J. Vis. Commun. Image R. 58 (2019) 334–344

Contents lists available at ScienceDirect

J. Vis. Commun. Image R.

journal homepage: www.elsevier.com/locate/jvci

High-capacity reversible data hiding in encrypted images based on extended run-length coding and block-based MSB plane rearrangement ☆

Kaimeng Chen [a], Chin-Chen Chang [b,*]

[a] Computer Engineering College, JiMei University, Xiamen 361021, China
[b] Department of Information Engineering and Computer Science, Feng Chia University, Taichung 40724, Taiwan

A R T I C L E I N F O

Article history:
Received 18 May 2018
Revised 11 October 2018
Accepted 8 December 2018
Available online 11 December 2018

Keywords:
Reversible data hiding
Run-length coding
Image encryption

A B S T R A C T

In this paper, we propose a novel reversible data hiding method in encrypted images. The proposed method takes full advantage of the spatial correlation in the original images to vacate room for embedding data before image encryption. By jointly using an extended run-length coding and a block-based most significant bit (MSB) plane rearrangement mechanism, the MSB planes of images can be compressed efficiently to generate room for high-capacity embedding. The receiver can extract data directly from encrypted images with only the data hiding key, and the original image or the high-quality plain image that contains secret data can be recovered with only the encryption key. The experimental results prove that the proposed method can reach a high embedding rate and a high PSNR.

1. Introduction

Reversible data hiding (RDH) is a technique that imperceptibly embeds secret data into the cover data, such as images, audios, videos, and texts, while the embedded data can be extracted and the cover data can be recovered by receivers [1]. The RDH technology is quite useful for some special applications in which images are not allowed to be disturbed, such as military, medical, and law forensics applications. To date, many RDH schemes have been proposed to embed secret data into digital images. These RDH schemes are based on include histogram shifting [2–4], difference expansion [5–7], and pixel value ordering [8–10]. Basically, these RDH schemes in plain images make use of the redundancy information to embed secret data in a reversible manner. If an image has been encrypted, the noise-like encrypted image loses a lot of the redundancy information. Therefore, these methods are not suitable for encrypted images.

With the development of cloud storing and computing, a large number of privacy data, such as videos and images, are stored and processed on cloud servers instead of clients' terminals. To protect privacy contents, users' data should be encrypted before being transmitted to cloud servers. To help the data management of encrypted images, RDH in encrypted images (RDHEI) has attracted the attention of many researchers. The RDHEI technology allows the data hider to embed data into encrypted images without knowing the content of the images. Subsequently, receivers can extract embedded data and recover the original image.

The first RDHEI method was proposed in [11]. Recently, Several RDHEI methods have been developed. Some of the methods embed secret data by modifying the pixels of encrypted images and use the spatial correlation of plaintext images to extract the embedded data after image decryption. Zhang [12] proposed a method to embed data by flipping three least significant bits (LSBs) of pixels in encrypted images. First, the encrypted image was divided into blocks of the same size, and each block was divided into two pixel-groups H_0 and H_1. To embed a bit of 0 or 1 into a block, three LSBs of all pixels in H_0 or H_1 of the block were flipped. After image decryption, for each block, the receiver flipped three LSBs of all pixels in H_0 and H_1 respectively to generate two blocks. The original block and the embedded bit can be identified between the two blocks by a smoothness estimation function. Based on [12], several improved RDHEI methods were proposed in [13–15]. In these studies, more precise functions were designed to estimate the smoothness of the decrypted images, thereby improving the accuracy of data extraction. Liao et al. [14] extended the smoothness function in [12] by taking the pixels in the borders of blocks into account. For each block to be recovered, if any recovered neighboring block

☆ This paper has been recommended for acceptance by Zicheng Liu.
* Corresponding author at: Department of Information Engineering and Computer Science, Feng Chia University, No. 100 Wenhwa Rd., Seatwen, Taichung 407, Taiwan.
E-mail address: alan3c@gmail.com (C.-C. Chang).

https://doi.org/10.1016/j.jvcir.2018.12.023
1047-3203/© 2018 Elsevier Inc. All rights reserved.

exists, the pixels in the adjacent border of the recovered block should be taken into account.

In the previous methods, data extraction and image recovery should be performed jointly. To separate the processes of data extraction and image recovery, separate RDHEI methods have been proposed. Zhang [16] proposed a separable RDHEI method based on LSB compression. The encrypted image was divided into pixel-groups. For each pixel-groups, the LSBs of all pixels were extracted and compressed to reserve embedding room. The secret data can be extracted directly from the embedding room, and image recovery can be performed without data extraction. In [17], half the fourth LSBs of encrypted pixels were compressed by low-density parity-check code (LDPC code) to provide room for embedding data. In [18], Slepian-Wolf source encoding was used to compress the MSBs of the encrypted image to reserve embedding space. In [19], the encrypted image was divided into blocks corresponding to original image's smooth and complex regions, and the LSBs of the blocks corresponding to smooth regions are compressed for embedding room. In [20], the blocks of the encrypted image were categorized into two sets according to their compressibility to run-length coding. The blocks of two sets are compressed respectively by run-length coding and matrix compression. In [21], the secret data were embedded into the pixels of the encrypted image by MSB substitution. The receiver can extract the secret data from the MSB plane of the encrypted image and recover the image by MSB prediction after decryption. In [22], a block-level predictor was adopted to generate prediction-error sequences from the encrypted image. Based on the prediction-error sequences, a difference expansion scheme was used to embed secret data. In [23], images were encrypted by public key cryptosystems with probabilistic and homomorphism properties, and data were embedded by multi-layer, wet-paper coding. In [24], before encryption, the image was divided into groups, and the groups were encrypted by RC4 with the same key. Since the encrypted groups maintain structure redundancy, a difference histogram is generated according to the encrypted image, and a histogram shifting method is performed for data hiding.

All of the published methods mentioned above involved vacating embedding room after image encryption. Since encryption would minimize the redundancy of images, it is difficult to achieve a satisfactory capacity of embedding space in encrypted images. To improve the embedding rate, some methods have been proposed that reserve room before image encryption (RRBE) [25–27]. In [25], to vacate embedding room in the LSB planes of the original image, a traditional RDH method was used to embed the LSBs of the selected pixels into other pixels. In [26], the original image was divided into patches, and each patch was compressed by using a sparse coding technology and an over-complete dictionary. In [27], the method divided the MSB planes of the original image into blocks of the same size. Each block was processed respectively according to its distribution of bits. If all bits in a block were identical, the block can be represented by only two bits. If only a few bits in a block were different, the block can be represented as the block type, the number of minority bits, and their locations by less bits. The spared room in the compressed blocks is used to accommodate the original LSBs. After encryption, the data hider can exploit the LSBs of the encrypted image to embed data.

For a plaintext image, spatial correlation exists in neighboring pixels. The neighboring bits in the MSB planes of the image are likely to be identical. Therefore, MSB planes can be transformed into bit streams containing long sequences of the same bits. Motivated from this, we have designed a run-length coding-based compression scheme to heavily compress the MSB planes of the original image for a large embedding capacity.

In this paper, we propose a novel RRBE separable RDHEI method with high capacity. First, we design a joint room vacating technique for plain images, which is composed of extended run-length coding and the block-based MSB plane rearrangement (BMPR) scheme. The joint technique can compress the MSB planes of images efficiently to achieve a lot of spare room. Based on the joint technique, an RDHEI method is proposed. First, the proposed method uses the joint scheme to perform lossless compression on the MSB planes of the plain image to make room, and, then, it embeds LSBs to the compressed MSB planes before encryption. The processed image is encrypted by using the bitwise XOR operation, and the LSBs of the image are reserved for the data hider. The proposed method allows the receiver to use different keys to extract the secret data, decrypt the image, and recover the original image from the embedded encrypted image.

The contributions of this paper are summarized as follows.

(1) We design a block-based MSB plane rearrangement (BMPR) scheme which transforms the MSB planes of the original image into high compressible bit streams, and an extended run-length coding which compresses the transformed bit streams at a high compression ratio.
(2) Based on MSB plane compression, we present an RRBE RDHEI method. The method can achieve high embedding capacity and high quality of the marked decrypted image. Data extraction and image recovery of the method are separable and error-free.

The rest of the paper is organized as follows. Section 2 introduces our run-length coding scheme and block-based MSB plane rearrangement scheme. The proposed RDHEI scheme is discussed in detail in Section 3. Section 4 provides the experimental results and comparisons. The conclusions are given in Section 5.

2. The joint embedding-room-vacating scheme

In this section, we propose a new joint scheme to vacate embedding room in the MSB planes of images. The joint scheme is composed of an extended run-length coding scheme and a block-based MSB plane rearrangement scheme. The basic idea of the joint scheme is to rearrange bits in the MSB planes to construct bit streams that have consecutive long sequences of $(0)_2$ or $(1)_2$ and compress them by extended run-length coding.

Any pixel in the images is constructed of eight bits. Thus, hereafter, the most significant bit of a pixel also is denoted as 8th-bit, and the least significant bit of a pixel is denoted as the1st-bit.

2.1. Extended run-length coding

Run-length coding is an efficient lossless data compression technique that is used to compress a consecutive sequence of the same symbols [28]. This type of sequence is called a "run." Run-length coding indicates a run by using two messages, i.e., the repeating symbol in the run, and the length of the run (i.e., the number of the symbol).

Due to the spatial correlation of images, the MSB planes of images have large numbers of long runs of 0 s and 1 s. Therefore, we extend the run-length coding to compress MSB planes to make room for embedding. In our extended run-length coding, there are two types of codewords:

(1) Fixed-length codeword: a fixed-length codeword consists of a prefix and a fixed number of tail bits. Fixed-length codewords are used to address runs with the length $L < 4$. The prefix of the codeword is fixed to bit 0. Beginning with the first bit of the run, a fixed number of sequential bits is extracted from the original bit stream as the number of tail

bits. Hereinafter, the fixed number is denoted as L_{fix}. The length of each fixed-length codeword is fixed to $L_{fix} + 1$. The L_{fix} is set in advance before performing compression procedure.

(2) Variable-length codeword: a variable-length codeword consists of a prefix, a length symbol and a tail bit. Variable-length codewords are used to compress runs with the length $L \geq 4$. For a codeword of a run with the length L, the length of the prefix $L_{pre} = \lfloor log_2 L \rfloor$, and the prefix consists of $L_{pre} - 1$ sequential bits 1 and ends with bit 0. The prefix indicates that $2^{L_{pre}} \leq L < 2^{L_{pre}+1}$. The length symbol also consists of L_{pre} bits, and it indicates the length $l = L - 2^{L_{pre}}$. The tail bit indicates the repeat bit 1 or 0. Therefore, the compression ratio of the run is $\frac{L}{2\lfloor log_2 L \rfloor + 1}$.

Fig. 1 shows an example of the compression procedure, where L_{fix} is set to 4. The bit stream is scanned bit by bit so that each run is extracted and encoded. First, the run of eleven 1 s is extracted and encoded as a variable-length codeword C_1. The prefix of C_1 is 110 which indicates $2^3 \leq 11 < 2^4$. The length symbol of C_1 is 011 which indicates the length $l = 11 - 2^3 = 3$. The tail bit is 1 which indicates the repeat bit. Therefore, the 11-bit run of 1 s is encoded as 7-bit codeword 1100111. Then, the next run of two 0 s is extracted. Since the length of the run is less than 4, the run is encoded as a fixed-length codeword C_2 with the prefix 0. Beginning with the first bit of the run of two 0 s, L_{fix} sequential bits are extracted from the bit stream as the tail bits of C_2. Therefore, the codeword C_2 is 00010. Finally, the last run of sixteen 0 s is encoded as 111000000.

We used fixed-length codewords to address runs with short length due to the existence of complex regions in images. If there are runs of short length in the region of the MSB plane, the region of the image is likely to be the complex region, and more short-length runs would exist in the next bits. Therefore, we extracted more bits to construct a relatively long tail to skip the unsmooth image area more quickly and reduce the expansion of the code size due to coding short runs.

2.2. Rearrangement of the block-based MSB plane

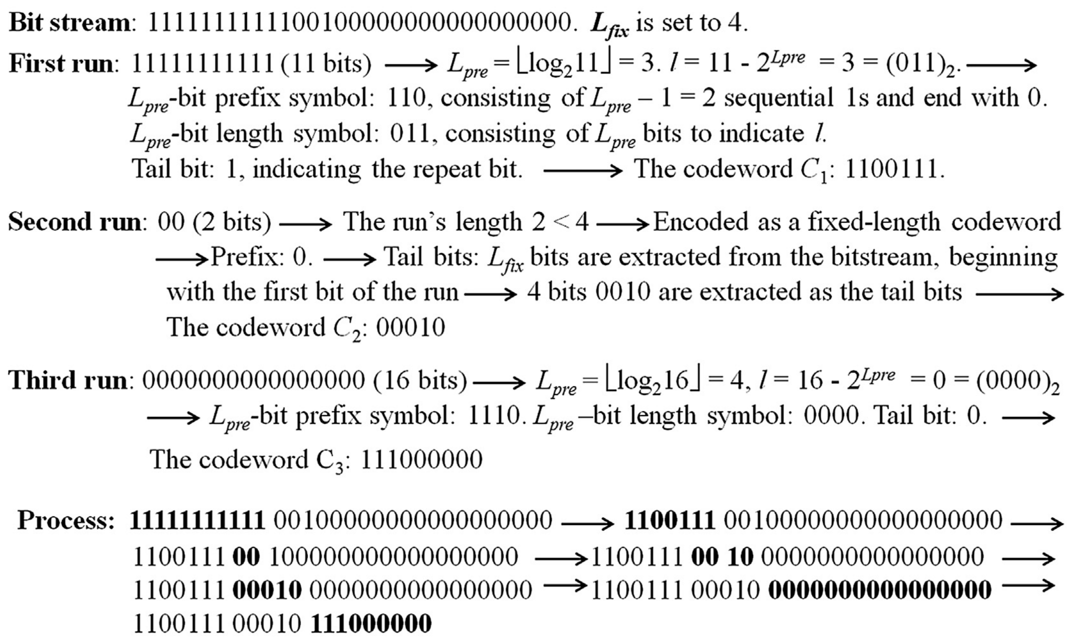
To get a higher compression rate of the MSB planes, a block-based MSB plane rearrangement (BMPR) scheme, which makes use of spatial correlation, was designed to construct the bit stream with longer length runs.

In the BMPR scheme, an image is divided into blocks of the same size, and then bits are extracted from the blocks one by one to construct the bit stream. The order of bits to be extracted from a block and the order of blocks to be extracted are shown in Table 1. Fig. 2 shows an example of different types of rearrangement in BMPR based on a 4×4 matrix and 2×2 block size.

Fig. 3 shows how the BMPR scheme affects the compression ratio of MSB planes. The 8×8 binary matrix is a part of the 8th bit-plane of Lena (Fig. 6(d)). If the matrix is compressed directly row by row or column by column by using extended run-length coding ($L_{fix} = 3$), the length of the code is 33. If the bits of the matrix are rearranged by the BMPR scheme (type 00 and 4×4 block size), the length of the code is 23. Due to spatial correlation, bits in the same block are likely to be identical, and bits in the adjacent borders of neighboring blocks are also likely to be the same. Therefore, if the bit stream is constructed block by block, the bit stream can contain runs with longer length. As shown in Fig. 3, compared with the row-by-row or column-by-column bit stream, the bit stream constructed by BMPR scheme can aggregate more bits of 1 into the same run and achieve higher compressibility.

2.3. Vacating room for embedding

Based on BMPR scheme and the extended run-length coding, the joint-embedding, room-vacating scheme can be performed to vacate room in LSBs for embedding data. The procedure of the joint scheme is given in Algorithm 1. First, for each bit-plane to be compressed, four bit streams are constructed according to four types of MSB plane rearrangement in Table 1. Then, the four bit streams are compressed respectively. The shortest compressed bit stream, the type and the length information are adopted to represent the original content of the bit-plane. Therefore, the MSB planes are com-


Bit stream: 11111111111001000000000000000000. L_{fix} is set to 4.

First run: 11111111111 (11 bits) $\longrightarrow$ $L_{pre} = \lfloor \log_2 11 \rfloor = 3$. $l = 11 - 2^{Lpre} = 3 = (011)_2$. $\longrightarrow$
L_{pre}-bit prefix symbol: 110, consisting of $L_{pre} - 1 = 2$ sequential 1s and end with 0.
L_{pre}-bit length symbol: 011, consisting of L_{pre} bits to indicate l.
Tail bit: 1, indicating the repeat bit. $\longrightarrow$ The codeword C_1: 1100111.

Second run: 00 (2 bits) $\longrightarrow$ The run's length $2 < 4$ $\longrightarrow$ Encoded as a fixed-length codeword $\longrightarrow$ Prefix: 0. $\longrightarrow$ Tail bits: L_{fix} bits are extracted from the bitstream, beginning with the first bit of the run $\longrightarrow$ 4 bits 0010 are extracted as the tail bits $\longrightarrow$
The codeword C_2: 00010

Third run: 0000000000000000 (16 bits) $\longrightarrow$ $L_{pre} = \lfloor \log_2 16 \rfloor = 4$, $l = 16 - 2^{Lpre} = 0 = (0000)_2$
$\longrightarrow$ L_{pre}-bit prefix symbol: 1110. L_{pre}–bit length symbol: 0000. Tail bit: 0. $\longrightarrow$
The codeword C_3: 111000000

Process: **11111111111** 001000000000000000000 $\longrightarrow$ **1100111** 001000000000000000000 $\longrightarrow$
1100111 **00** 1000000000000000000 $\longrightarrow$ 1100111 **00 10** 0000000000000000 $\longrightarrow$
1100111 **00010** 0000000000000000 $\longrightarrow$ 1100111 00010 **0000000000000000** $\longrightarrow$
1100111 00010 **111000000**

Fig. 1. Example of compression procedure.

Table 1
Four types of MSB plane rearrangement.

Type label	The order of bits inside the block	The order of blocks
00	Row by row	Row by row
01	Row by row	Column by column
10	Column by column	Row by row
11	Column by column	Column by column

pressed. Finally, the LSBs are embedded to the spared room of the compressed bit-planes to vacate room for embedding. For an image sized $m \times n$, the BMPR scheme scans each bit of the bit plane once to construct the bit stream, and the extended run-length coding scans each bit of the bit stream once to compress the bit stream. Therefore, to vacate room in an image sized $m \times n$, the computation complexity of the algorithm is O(mn).

Algorithm 1 (*The joint embedding-room-vacating scheme*).

```
Input: image $I$, block size $s \times s$, fixed-length $L_{fix}$
1:  Extract the bit-planes of 8th-bit, 7th-bit, 6th-bit, 5th-bit and 4th-bit from $I$;
2:  for each plane from 8th bit-plane to 4th bit-plane do
3:      Divide the bit-plane into blocks with the size of $s \times s$;
4:      Construct four types of bit streams, i.e., $BS_{00}$, $BS_{01}$, $BS_{10}$, and $BS_{11}$ from the bit-plane by using BMPR;
5:      Compress $BS_{00}$, $BS_{01}$, $BS_{10}$, and $BS_{11}$, respectively, by extended run-length coding with $L_{fix}$;
6:      Select the shortest compressed bit stream $CBS$ from the compressed $BS_{00}$, $BS_{01}$, $BS_{10}$, and $BS_{11}$;
7:      if the length of $CBS$ is shorter than the original $BS_{ij}$
8:          Adopt the type, length, and content of $CBS$ to represent the bit-plane;
9:      else
10:         Do not compress the plane and discontinue overall compression process;
11:     end if
12: end for
13: Record block size, $L_{fix}$, and the number of compressed bit-planes in the 8th bit-plane.
14: Embed the LSBs of $I$ to the spared room of the compressed bit planes to vacate embedding room in LSB planes.
15: Record the available capacity in the 1st bit-plane;
Output: a room-vacated image, $RI$
```

Fig. 4 shows an example of vacating room. After compression, for the 8th bit-plane, the first 6 bits are used to record the block size and L_{fix}. Then the next 3 bits are used to record the number of compressed bit planes. Finally, 2-bit BMPR type, 6-bit length information and the content of the compressed bit stream are recorded. For the 7th bit-plane, it is compressed into 2-bit BMPR type, 6-bit length information and the content of the compressed bit stream. The original bits of the two compressed bit-planes can be perfectly recovered. The spared 27 bits of the 8th bit-plane and 31 bits of the 7th bit-plane are used to accommodate 58 bits of LSB plane. Therefore, the embedding room is vacated in LSB plane, and the capacity of the room is recorded in the first 6 bits of the LSB plane.

As shown in Fig. 4, the embedding capacity depends on the compression ratios of the MSB planes. For the run-length coding, compression ratio increases with the length of the run. When image size increases, the constructed bit stream contains longer runs due to the increase of bits in the bit-plane. Therefore, images with larger size can achieve higher compression ratios and provide better embedding capacity. For images with the same size, smooth images have stronger spatial correlation than complex images. Bit streams constructed from smooth images contain longer runs. Therefore, smooth images can provide better embedding capacity than complex images.

Based on the joint embedding-room-vacating scheme, the data owner can make room for the data hider to embed data in encrypted images. The RDHEI method images is explained in Section 3.

3. The proposed RDHEI method

Fig. 5 shows an overview of the proposed RDHEI method. First, the content owner vacates room for embedding data in the original image. Then, the owner encrypts and shuffles the vacated image using the encryption key and the shuffle key. In the data-embedding phase, the data hider encrypts the secret data using a data hiding key. Then, the data hider reshuffles the encrypted image and embeds the secret data in its LSBs and shuffles the embedded image again with the same shuffle key. The receiver can recover the image or extract secret data independently by using different keys. Table 2 provides a list of the security keys in the proposed method.

3.1. Image encryption and shuffle

Assume that the size of original image I is $N_1 \times N_2$ and that the gray value of its pixel $I(i, j)$ falls into [0,255] ($i \in [1, N_1], j \in [1, N_2]$). The gray value of each pixel $I(i, j)$ is denoted as V_{ij}, and the eight bits of V_{ij} are denoted as $b_{ij1}, b_{ij2}, \ldots, b_{ij8}$. The values of b_{ijk} and V_{ij} are calculated as follows:

$$b_{ijk} = \lfloor \frac{V_{ij}}{2^{k-1}} \rfloor \bmod 2, \qquad k = 1, 2, ..., 8. \tag{1}$$

$$V_{ij} = \sum_{k=1}^{8} (2^{k-1} \times b_{ijk}). \tag{2}$$

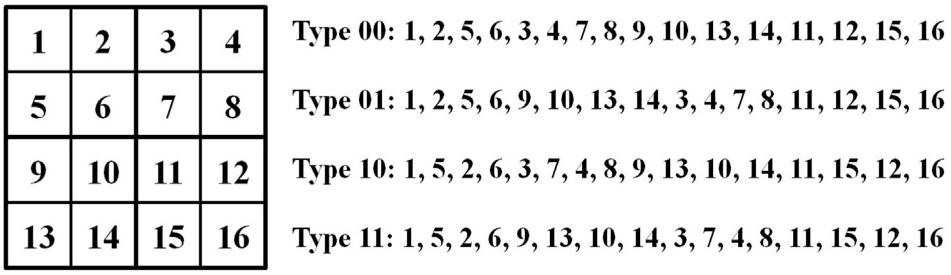

1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
Type 00: 1, 2, 5, 6, 3, 4, 7, 8, 9, 10, 13, 14, 11, 12, 15, 16
Type 01: 1, 2, 5, 6, 9, 10, 13, 14, 3, 4, 7, 8, 11, 12, 15, 16
Type 10: 1, 5, 2, 6, 3, 7, 4, 8, 9, 13, 10, 14, 11, 15, 12, 16
Type 11: 1, 5, 2, 6, 9, 13, 10, 14, 3, 7, 4, 8, 11, 15, 12, 16

Fig. 2. Different types of rearrangement in the BMPR scheme.

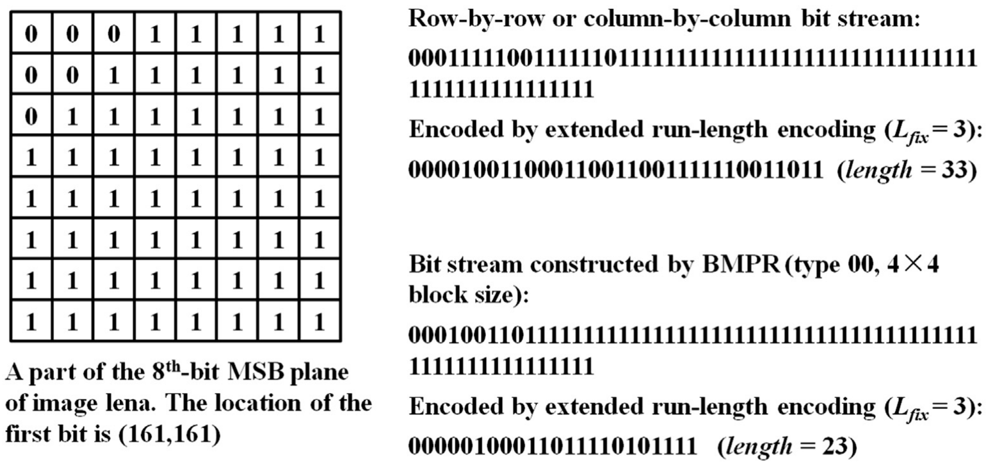

Row-by-row or column-by-column bit stream:
0001111100111111011111111111111111111111111111111111111111111111
Encoded by extended run-length encoding (L_{fix} = 3):
000010011000110011001111110011011 (length = 33)
Bit stream constructed by BMPR (type 00, 4×4 block size):
0001001101111111111111111111111111111111111111111111111111111111
Encoded by extended run-length encoding (L_{fix} = 3):
00000100011011110101111 (length = 23)
A part of the 8th-bit MSB plane of image lena. The location of the first bit is (161,161)

Fig. 3. Affect of the BMPR scheme on the compression of the MSB planes of images.

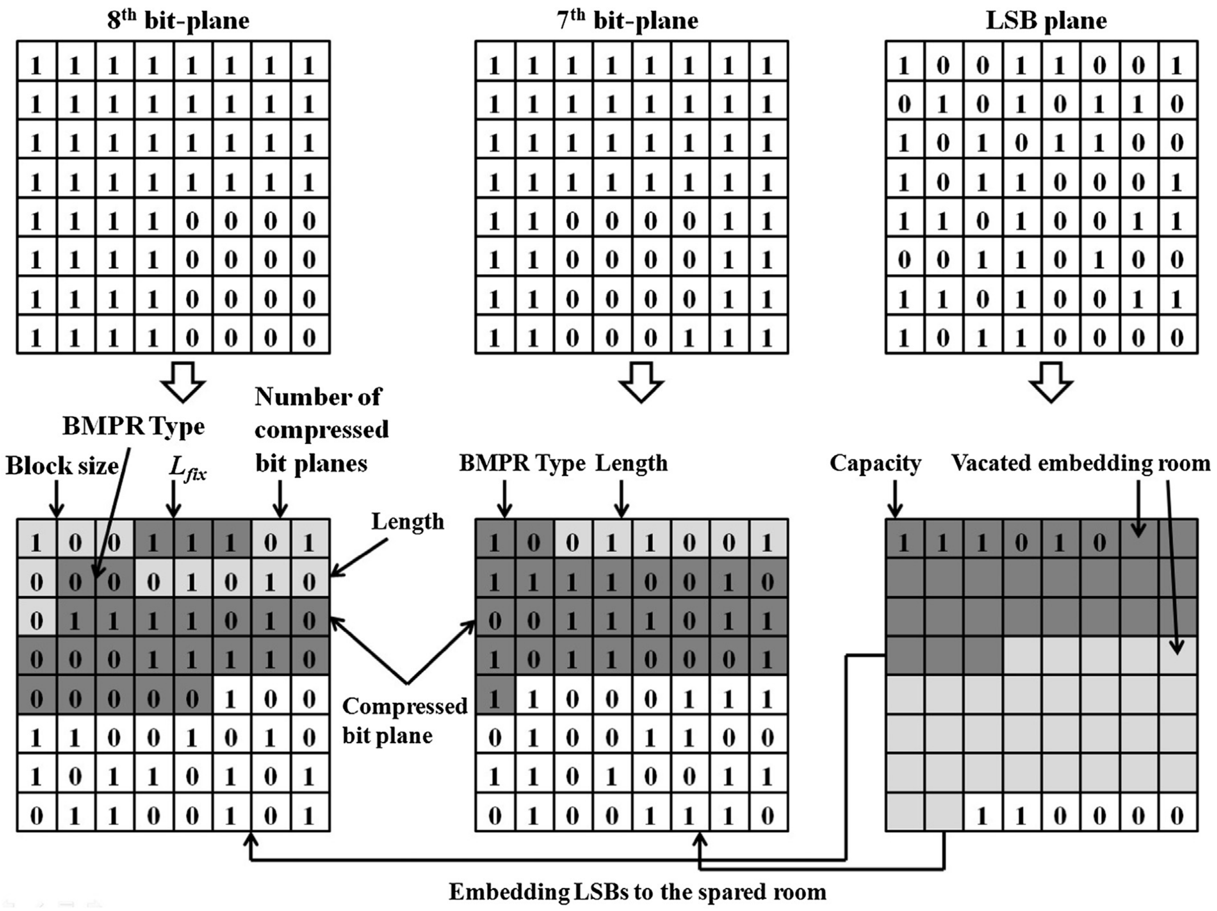


Fig. 4. Example of vacating room.

Before encryption, the joint embedding-room-vacating scheme, as proposed in Section 2, is used to process the original image I to generate a room-vacated image RI. The gray value of each pixel in RI is denoted as V'_{ij}, and the eight bits of V'_{ij} are denoted as b'_{ij1}, $b'_{ij2}, \ldots, b'_{ij8}$.

A stream cipher is used to generate the random bit stream, $\{e_{ij0}, e_{ij2}, \cdots, e_{ij8}\}_{N1\times N2}$, according to the encryption key, K_{en}. Then, the encrypted image, RI_e, is generated as:

$$B_{ijk} = \begin{cases} b'_{ijk}, & for\ bits\ indicating\ embedding\ capacity \\ b'_{ijk} \oplus e_{ijk}, & for\ the\ other\ bits \end{cases} \tag{3}$$

Considering that the information of embedding capacity is not encrypted, to protect the capacity information and enhance security, an encrypted image shuffle scheme is used in the proposed

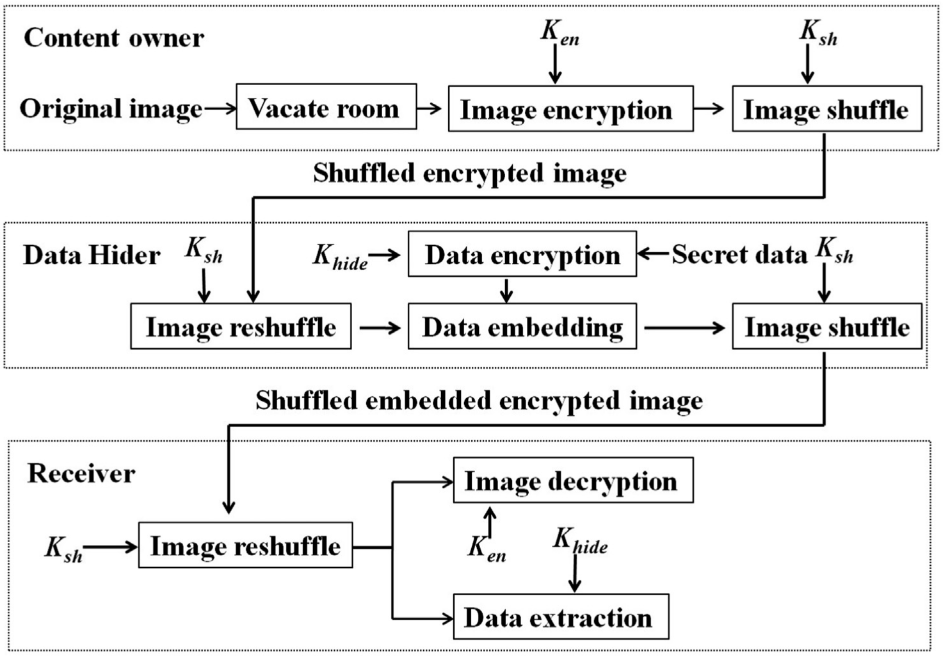


Fig. 5. Overview of the proposed method.

Table 2
The list of security keys used in the proposed method.

Notation	Key	Owner	Purpose
K_{en}	Encryption key	Content owner, receiver	Encrypt and decrypt the original image
K_{sh}	Shuffle key	Content owner, data hider, receiver	Shuffle and reshuffle the encrypted image
K_{hide}	Data hiding key	Data hider, receiver	Select substitutable LSBs in the encrypted image for embedding secret bits

Table 3
Average embedding rates of the proposed method with different blocks and two competitors on 1000 images.

	Average embedding rate (bpp)		Average embedding rate (bpp)
Proposed (2 × 2)	2.4355	Proposed (16 × 16)	2.2156
Proposed (4 × 4)	2.4248	Li [24]	0.6899
Proposed (8 × 8)	2.3000	BBE [27]	2.2341

method. According to the shuffle key K_{sh}, the content owner shuffles all pixels in RI_e to generate the shuffled encrypted image RI_{se}. Finally, RI_{se} is sent to the data hider for embedding data.

3.2. Data embedding

Since the proposed method vacates room in LSB planes by embedding LSBs to the compressed MSB planes, the secret data are embedded into LSB planes by LSB substitution. To embed data, the data hider first reshuffles the received RI_{se} to the encrypted image RI_e by the shuffle key K_{sh}. Then, the capacity information can be extracted from the 1st bit plane of RI_e to obtain all substitutable bits in LSB planes. By the data hiding key K_{hide}, the data hider pseudo-randomly selects the substitutable bits and substitutes them with the secret data. After embedding, the embedded encrypted image RI_{ee} is shuffled again by the shuffle key K_{sh} to generate the shuffled embedded encrypted image RI_{see}.

3.3. Recovery of the image and extraction of data

When the receiver accepts the shuffled embedded encrypted image RI_{see} from the data hider, the receiver can extract data from the encrypted image RI_{see} without decryption by using K_{hide} and K_{sh}. If the receiver has K_{en} and K_{sh}, the receiver can recover the original image or a highly similar image with the embedded data.

(1) *Data extraction*. The receiver who has K_{hide} and K_{sh} can extract the embedded data from RI_{see} directly without decryption. First, by using K_{sh}, the receiver reshuffles RI_{see} to get the embedded encrypted image RI_{ee}. Then, by using

Table 4
Average compression ratio of each compressed bit-plane using the proposed method.

Blocks size	8th bit-plane	7th bit-plane	6th bit-plane	5th bit-plane	4th bit-plane
2 × 2	7.2993	3.0012	1.8769	1.4059	1.1762
4 × 4	7.3910	2.9994	1.8653	1.3968	1.1703
8 × 8	6.7476	2.7685	1.7553	1.3410	1.1427
16 × 16	6.2228	2.6028	1.6892	1.3122	1.1296

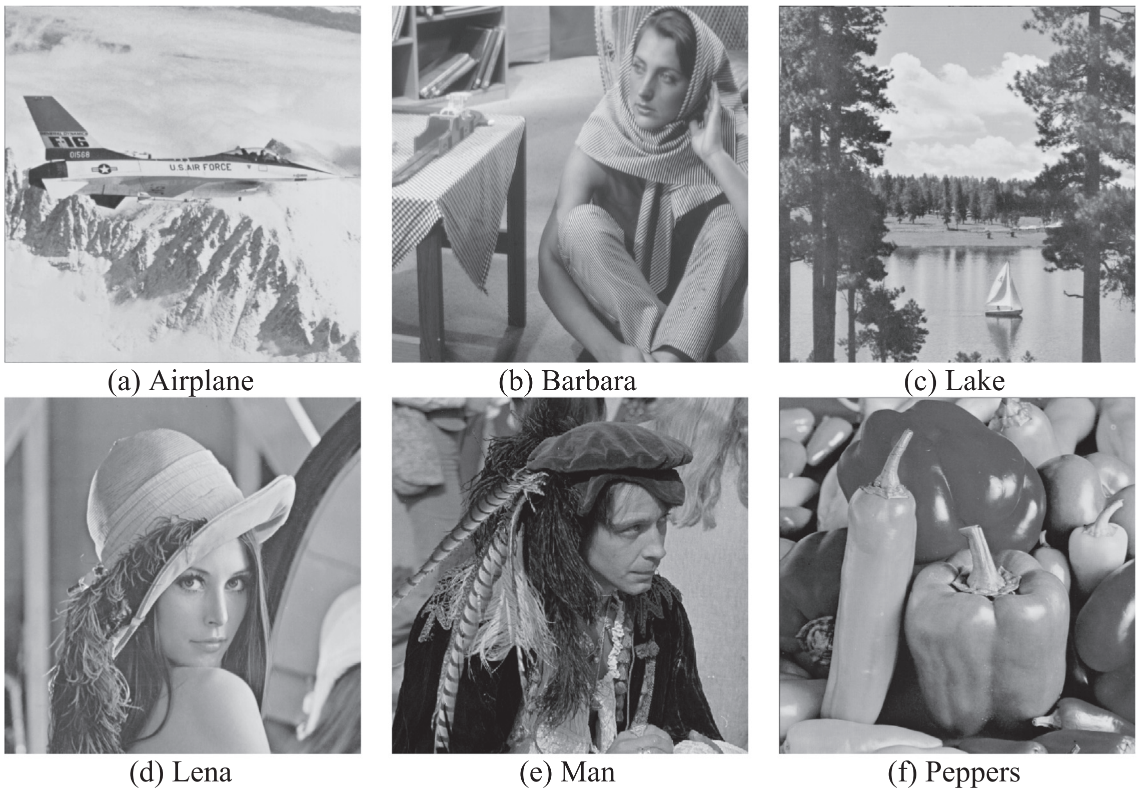


Fig. 6. Six test images.

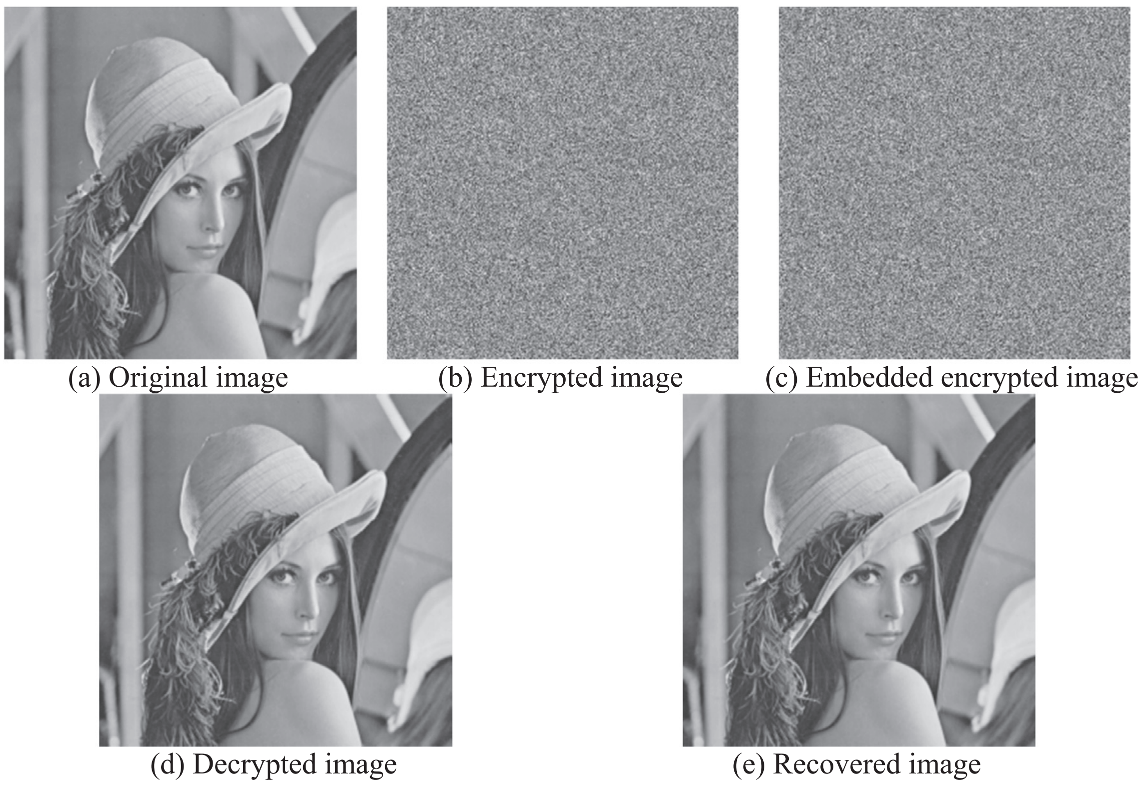


Fig. 7. Different images of Lena generated by the proposed method.

K_{hide}, the receiver can directly extract each bit of the embedded secret data from the LSB planes of RI_{ee} to obtain the original secret data.

(2) *Image recovery*. The receiver who has K_{en} and K_{sh} can recover the original image or a highly similar image with the embedded data. First, the receiver reshuffles RI_{see} into the embed-

Table 5
Encrypted images' PSNRs and correlation coefficients with the plain images.

Encrypted image	Airplane	Barbara	Lake	Lena	Man	Peppers
PSNR	8.0463	8.8217	8.3395	9.2296	7.9520	8.4474
Correlation coefficient	−0.0003	0.0006	0.0005	0.0007	−0.0011	0.0009

Table 6
Embedding rate, PSNR and MSE of the proposed method under various block sizes.

(a)

	Capacity (bpp)		MSE		PSNR (dB)	
Block size	2×2	4×4	2×2	4×4	2×2	4×4
Airplane	2.3187	2.3407	5.0583	5.2302	41.0908	40.9456
Barbara	1.3991	1.4091	1.3002	1.3224	46.9907	46.9171
Lake	1.5290	1.5561	1.5575	1.6156	46.2065	46.0474
Lena	1.9725	1.9444	2.4429	2.3821	44.2517	44.3612
Man	1.6833	1.6787	1.8338	1.8238	45.4973	45.5211
Peppers	1.8562	1.8796	2.2152	2.2665	44.6767	44.5772

(b)

	Capacity (bpp)		MSE		PSNR (dB)	
Block size	8×8	16×16	8×8	16×16	8×8	16×16
Airplane	2.1625	2.0270	3.8144	2.7185	42.3165	43.7876
Barbara	1.2781	1.1569	1.0598	0.8179	47.8787	49.0040
Lake	1.4061	1.2932	1.3112	1.0810	46.9542	47.7926
Lena	1.8557	1.7534	2.2134	2.0091	44.6802	45.1007
Man	1.5544	1.4590	1.5787	1.4000	46.1478	46.6695
Peppers	1.7256	1.5635	1.9509	1.6277	45.2284	46.0151

Table 7
Average compression ratio of each image under various block sizes.

Block size	Airplane	Barbara	Lake	Lena	Man	Peppers
2×2	1.8647	1.3886	1.4405	1.6516	1.5075	1.5904
4×4	1.8802	1.3924	1.4518	1.6363	1.5055	1.6024
8×8	1.7621	1.3434	1.3912	1.5902	1.4511	1.5270
16×16	1.6818	1.3010	1.3489	1.5401	1.4120	1.4550

Table 8
Comparison of the embedding rate (bpp).

	Airplane	Barbara	Lake	Lena	Man	Peppers
Li [24]	0.7389	0.6144	0.7379	0.7752	0.6939	0.7510
BBE [27]	2.2043	1.3198	1.5351	1.8195	1.6070	1.8206
Proposed	2.3407	1.4091	1.5561	1.9725	1.6833	1.8796

ded encrypted image RI_{ee}. By using K_{en}, RI_{ee} is decrypted into the room-vacated image with the embedded data RI_e. After decryption, block size, L_{fix} and the number of the compressed bit-planes can be extracted from the 8th bit-plane. For each compressed bit plane, the compressed bit stream and the embedded LSBs are extracted and separated. According to block size, L_{fix} and BMPR type, the compressed bit stream is decompressed and rearranged to recover the compressed bit plane. Since the extended run-length coding is a lossless compression scheme, the compressed bit planes are recovered without any error.

If the embedded LSBs are not filled back to the embedding room, a marked decrypted image is generated. Since the MSB planes of the image are recovered perfectly, the marked decrypted image is very similar to the original image. If the embedded LSBs are filled back to the embedding room, the LSB planes of the decrypted image are recovered to their original situations, and the recovered image is completely the same as the original image. Therefore, the proposed method is an error-free reversible method.

4. Experimental results

In this section, we conduct experiments to evaluate the performance of the proposed method, and perform comparisons between the proposed method and several previous RDHEI methods.

First, we evaluate the embedding rates of the proposed method with different block sizes (the length of fixed-length codewords is set to 9), and results are compared with two competitors [24] and [27]. The experiments are performed on 1000 $512 \times 512 \times 8$ grayscale images randomly selected from BOSSBase [29].

Table 3 shows the average embedding rate of each method. When block size is 2×2 or 4×4, the proposed method achieves higher embedding rate and outperforms the competitors. Table 4 shows the average compression ratio of each compressed bit-plane using the extended run-length coding of the proposed

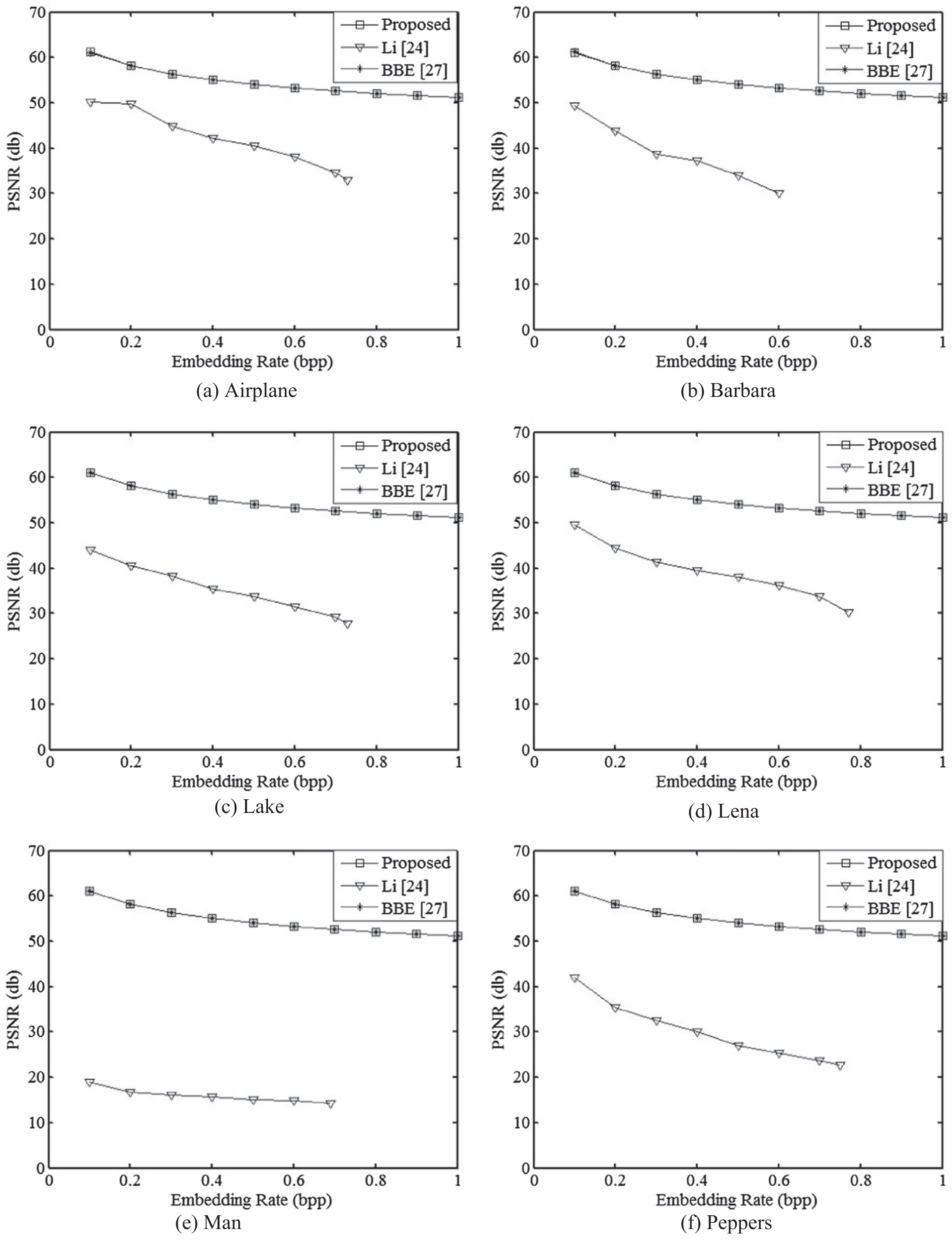


Fig. 8. Comparisons of PSNR values for the marked decrypted images generated by the proposed methods, Li [24] and BBE [27].

method. As shown in the table, when block size is small, the proposed method can achieve higher compression ratios. That results in higher embedding rates. When the block is too large, bits in the block are more likely to be different. Therefore, the runs shorten in the bit stream constructed by BMPR scheme, and the compressibility of the bit stream is reduced.

Fig. 6 shows six $512 \times 512 \times 8$ grayscale standard test images from [30]. We perform the proposed methods on the six images separately to evaluate the performance.

Fig. 7 shows the encrypted and decrypted images of the Lena image that were generated by the proposed method with the embedding rate, *R*, of 1.00 bit per pixel. The encrypted images shown in Fig. 7(b) and (c) are noise-like images, and that means they have a high perceptual security level. Fig. 7(d) is the decrypted image that contains the encrypted secret data, in which the PSNR value was 51.14 dB. Fig. 7(e) is the original recovered decrypted image, which is completely the same as the original image in Fig. 7(a). There are no visible differences in Fig. 7(d) and (e), which means the secret data are imperceptible.

Table 5 shows each encrypted image's PSNR value and correlation coefficient with each plain image. After image encryption, the PSNR of each encrypted image is very low so it's difficult to detect the content of the original image from its encrypted version. The correlation coefficients between encrypted images and the original images are almost zero, so no information can be achieved from the encrypted images without encryption key. For each $512 \times 512 \times 8$ test image, since every bit of the image is encrypted by a randomly generated bit, the encryption key space of the proposed scheme is $2^{512\times512\times8}$, which is large enough to ensure computational security. Therefore, the encryption scheme of the proposed method securely protects the privacy of the original image.

Table 6 shows the performance of the proposed method under different block sizes, including the embedding rate and the MSE/PSNR of the corresponding decrypted image. As shown in the table, different images should choose different block sizes to achieve maximum embedding capacity. Besides, image texture complexity affects embedding capacity significantly. Images with smooth textures (Airplane, Lena, Peppers) can provide larger embedding rate than images with complex textures (Barbara, Lake, Man). This is because images with smooth textures can form bit streams with longer runs due to stronger spatial correlation, resulting in higher compressibility.

The average compression ratios of each image under different block sizes are shown in Table 7. The results in Table 7 validate that images with smooth textures (Airplane, Lena, Peppers) achieve higher compression ratios.

Table 8 lists the embedding rate comparisons among the proposed scheme and two competitors [24,27]. The results in Table 8 show that the proposed scheme had better performance than the other methods in terms of embedding data payload. There are two reasons the proposed method was able to achieve a higher embedding rate. First, the proposed method vacates embedding room in plain images before image encryption, so the method can take full advantage of the redundancy of images. Second, the block-based MSB plane rearrangement scheme can construct bit streams that have more long runs for the extended run-length coding, allowing then scheme to attain high compression rate bit streams due to the spatial correlation of the images.

Fig. 8 compares the PSNR values of the marked decrypted images generated by different RDHEI methods. The results in Fig. 8 indicate that the proposed method can achieve relatively high PSNR values. The PSNR values of the proposed method are higher than 60 dB when the embedding rate is 0.1 bpp and higher than 50 db when the embedding rate is 1 bpp. This is because the proposed method uses only LSB planes to embed secret data. Compared with the two competitors in [24] and [27], the proposed method outperforms the method in [24], and has almost the same performance as the method in [27]. Since the method in [27] also embedded data in LSB planes, it has almost the same PSNR values as the proposed method.

5. Conclusions

In this paper, we propose a new high-capacity reversible data hiding method. The method adopts an extended run-length coding scheme and a block-based MSB plane rearrangement scheme to efficiently compress the MSB planes of the plain images before image encryption, so a large embedding room can be achieved. The hidden data can be extracted completely without error by using only the data hiding key. The original image can be recovered correctly from the decrypted image by using only the encryption key. The comparison of experimental results prove that the proposed method can achieve a much higher capacity of embedding room and a higher PSNR than previous methods.

Acknowledgement

This paper is supported by the Natural Science Foundation of Fujian Province, China (2017J05104), and the National Natural Science Foundation of China (61701191).

References

[1] Y.Q. Shi, Z. Ni, D. Zou, C. Liang, G. Xuan, Lossless data hiding: fundamentals, algorithms and applications, Proceedings of IEEE International Symposium on Circuits and Systems, Vancouver, Canada, vol. 2, 2004, pp. 33–36.

[2] Z. Ni, Y.Q. Shi, N. Ansari, W. Su, Reversible data hiding, IEEE Trans. Circ. Syst. Video Technol. 16 (3) (2006) 354–362.

[3] G. Coatrieux, W. Pan, N. Cuppens-Boulahia, F. Cuppens, C. Roux, Reversible watermarking based on invariant image classification and dynamical error histogram shifting, IEEE Trans. Inf. Forensics Secur. 8 (1) (2013) 111–120.

[4] T.S. Nguyen, C.C. Chang, N.T. Huynh, A novel reversible data hiding scheme based on difference-histogram modification and optimal EMD algorithm, J. Vis. Commun. Image Repres. 33 (2015) 389–397.

[5] J. Tian, Reversible data embedding using a difference expansion, IEEE Trans. Circ. Syst. Video Technol. 13 (8) (2003) 890–896.

[6] Y. Hu, H.K. Lee, J. Li, DE-based reversible data hiding with improved overflow location map, IEEE Trans. Circ. Syst. Video Technol. 19 (2) (2009) 250–260.

[7] Y. Qiu, Z. Qian, L. Yu, Adaptive reversible data hiding by extending the generalized integer transformation, IEEE Sign. Process Lett. 23 (1) (2016) 130–134.

[8] X. Li, J. Li, B. Li, B. Yang, High-fidelity reversible data hiding scheme based on pixel-value-ordering and prediction-error expansion, Sign. Process. 93 (1) (2013) 198–205.

[9] X. Qu, H.J. Kim, Pixel-based pixel value ordering predictor for high-fidelity reversible data hiding, Sign. Process. 111 (2015) 249–260.

[10] B. Ou, X. Li, J. Wang, High-fidelity reversible data hiding based on pixel-value-ordering and pairwise prediction-error expansion, J. Vis. Commun. Image Repres. 39 (Aug.) (2016) 12–23.

[11] W. Puech, M. Chaumont, O. Strauss, A reversible data hiding method for encrypted images, Proceedings of SPIE, San Jose, USA vol. 6819 (2008), pp. 68 191E-1–68 191E-9.

[12] X. Zhang, Reversible data hiding in encrypted images, IEEE Sign. Process Lett. 18 (4) (2011) 255–258.

[13] W. Hong, T. Chen, H. Wu, An improved reversible data hiding in encrypted images using side match, IEEE Sign. Process Lett. 19 (4) (2012) 199–202.

[14] X. Liao, C. Shu, Reversible data hiding in encrypted images based on absolute mean difference of multiple neighboring pixels, J. Vis. Commun. Image Repres. 28 (2015) 21–27.

[15] C. Qin, X. Zhang, Effective reversible data hiding in encrypted image with privacy protection for image content, J. Vis. Commun. Image Repres. 31 (2015) 154–164.

[16] X. Zhang, Separable reversible data hiding in encrypted image, IEEE Trans. Inf. Forens. Secur. 7 (2) (2012) 826–832.

[17] X. Zhang, Z. Qian, G. Feng, Y. Ren, Efficient reversible data hiding in encrypted images, J. Vis. Commun. Image Repres. 25 (2) (2014) 322–328.

[18] Z. Qian, X. Zhang, Reversible data hiding in encrypted image with distributed source encoding, IEEE Trans. Circuits Syst. Video Technol. 26 (4) (2016) 636–646.

[19] C. Qin, W. Zhang, F. Cao, X. Zhang, C.C. Chang, Separable reversible data hiding in encrypted images via adaptive embedding strategy with block selection, Sign. Process. 153 (2018) 109–122.

[20] C. Qin, Z. He, X. Luo, J. Dong, Reversible data hiding in encrypted image with separable capability and high embedding capacity, Inf. Sci. 465 (2018) 285–304.

[21] P. Puteaux, W. Puech, An Efficient MSB Prediction-Based Method for High-Capacity Reversible Data Hiding in Encrypted Images, IEEE Trans. Inform. Forens. Secu. 13 (7) (2018) 1670–1681.

[22] S. Yi, Y. Zhou, Z. Hua, Reversible data hiding in encrypted images using adaptive block-level prediction-error expansion, Sign. Process. Image Commun. 64 (2018) 78–88.

[23] X. Zhang, J. Long, Z. Wang, H. Cheng, Lossless and reversible data hiding in encrypted images with public key cryptography, IEEE Trans. Circ. Syst. Video Technol. 26 (9) (2016) 1622–1631.

[24] M. Li, D. Xiao, Y. Zhang, H. Nan, Reversible data hiding in encrypted images using cross division and additive homomorphism, Sign. Process. Image Commun. 39 (Part A) (2015) 234–248.

[25] K. Ma, W. Zhang, X. Zhao, N. Yu, F. Li, Reversible data hiding in encrypted images by reserving room before encryption, IEEE Trans. Inf. Forens. Secur. 8 (3) (2013) 553–562.
[26] X. Cao, L. Du, X. Wei, D. Meng, X. Guo, High capacity reversible data hiding in encrypted images by patch-level sparse representation, IEEE Trans. Cybern. 46 (5) (2016) 1132–1143.
[27] S. Yi, Y. Zhou, Binary-block embedding for reversible data hiding in encrypted images, Sign. Process. 133 (2017) 40–51.
[28] A. Chandra, K. Chakrabarty, Frequency-directed run-length (FDR) codes with application to system-on-a-chip test data compression, in: Proceedings of VLSI Test Symposium, 2001, pp. 42–47.
[29] P. Bas, T. Filler, T. Pevny, Break our steganographic system': the ins and outs of organizing BOSS, in: Proceedings of 13th International Workshop on Information Hiding, Prague, Czech Republic, 2011, pp. 59–70.
[30] Computer Vision Group, Test Image database. Online <http://decsai.ugr.es/cvg/dbimagenes/g512.php>.