Reversible Data Hiding in Encrypted Images by Reserving Room Before Encryption

Kede Ma, Weiming Zhang, Xianfeng Zhao, *Member, IEEE*, Nenghai Yu, and Fenghua Li

***Abstract*—Recently, more and more attention is paid to reversible data hiding (RDH) in encrypted images, since it maintains the excellent property that the original cover can be losslessly recovered after embedded data is extracted while protecting the image content's confidentiality. All previous methods embed data by reversibly vacating room from the encrypted images, which may be subject to some errors on data extraction and/or image restoration. In this paper, we propose a novel method by reserving room before encryption with a traditional RDH algorithm, and thus it is easy for the data hider to reversibly embed data in the encrypted image. The proposed method can achieve real reversibility, that is, data extraction and image recovery are free of any error. Experiments show that this novel method can embed more than 10 times as large payloads for the same image quality as the previous methods, such as for PSNR $= 40$ dB.**

***Index Terms*—Reversible data hiding, image encryption, privacy protection, histogram shift.**

I. Introduction

REVERSIBLE data hiding (RDH) in images is a technique, by which the original cover can be losslessly recovered after the embedded message is extracted. This important technique is widely used in medical imagery, military imagery and law forensics, where no distortion of the original cover is allowed. Since first introduced, RDH has attracted considerable research interest.

In theoretical aspect, Kalker and Willems [1] established a rate-distortion model for RDH, through which they proved the rate-distortion bounds of RDH for memoryless covers and proposed a recursive code construction which, however, does not approach the bound. Zhang *et al.* [2], [3] improved the recursive code construction for binary covers and proved that this construction can achieve the rate-distortion bound as long as the compression algorithm reaches entropy, which establishes the equivalence between data compression and RDH for binary covers.

In practical aspect, many RDH techniques have emerged in recent years. Fridrich *et al.* [4] constructed a general framework for RDH. By first extracting compressible features of original cover and then compressing them losslessly, spare space can be saved for embedding auxiliary data. A more popular method is based on difference expansion (DE) [5], in which the difference of each pixel group is expanded, e.g., multiplied by 2, and thus the least significant bits (LSBs) of the difference are all-zero and can be used for embedding messages. Another promising strategy for RDH is histogram shift (HS) [6], in which space is saved for data embedding by shifting the bins of histogram of gray values. The state-of-art methods [7]–[11] usually combined DE or HS to residuals of the image, e.g., the predicted errors, to achieve better performance.

With regard to providing confidentiality for images, encryption [12] is an effective and popular means as it converts the original and meaningful content to incomprehensible one. Although few RDH techniques in encrypted images have been published yet, there are some promising applications if RDH can be applied to encrypted images. In [13], Hwang *et al.* advocated a reputation-based trust-management scheme enhanced with data coloring (a way of embedding data into covers) and software watermarking, in which data encryption and coloring offer possibilities for upholding the content owner's privacy and data integrity. Obviously, the cloud service provider has no right to introduce permanent distortion during data coloring into encrypted data. Thus, a reversible data coloring technique based on encrypted data is preferred. Suppose a medical image database is stored in a data center, and a server in the data center can embed notations into an encrypted version of a medical image through a RDH technique. With the notations, the server can manage the image or verify its integrity without having the knowledge of the original content, and thus the patient's privacy is protected. On the other hand, a doctor, having the cryptographic key, can decrypt and restore the image in a reversible manner for the purpose of further diagnosing.

Some attempts on RDH in encrypted images have been made. In [16], Zhang divided the encrypted image into several blocks. By flipping 3 LSBs of the half of pixels in each block, room can be vacated for the embedded bit. The data extraction and image recovery proceed by finding which part has been flipped in one block. This process can be realized with the help of spatial correlation in decrypted image. Hong *et al.* [17] ameliorated Zhang's method at the decoder side by further exploiting the spatial correlation using a different estimation equation and side match technique to achieve much lower error rate. These two

Manuscript received July 30, 2012; revised February 12, 2013; accepted February 13, 2013. Date of publication February 25, 2013; date of current version March 07, 2013. This work was supported in part by the Natural Science Foundation of China under Grant 61170234 and Grant 60803155, and in part by the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant XDA06030601. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Z. Jane Wang.

K. Ma, W. Zhang, and N. Yu are with the School of Information Science and Technology, University of Science and Technology of China, Hefei, 230026, China (e-mail: k29ma@uwaterloo.ca; weimingzhang@yahoo.cn; ynh@ustc.edu.cn).

X. Zhao and F. Li are with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, 100093, China (e-mail: zhaoxianfeng@iie.ac.cn; lifenghua@iie.ac.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2013.2248725

1556-6013/$31.00 © 2013 IEEE

methods mentioned above rely on spatial correlation of original image to extract data. That is, the encrypted image should be decrypted first before data extraction.

To separate the data extraction from image decryption, Zhang [18] emptied out space for data embedding following the idea of compressing encrypted images [14], [15]. Compression of encrypted data can be formulated as source coding with side information at the decoder [14], in which the typical method is to generate the compressed data in lossless manner by exploiting the syndromes of parity-check matrix of channel codes. The method in [18] compressed the encrypted LSBs to vacate room for additional data by finding syndromes of a parity-check matrix, and the side information used at the receiver side is also the spatial correlation of decrypted images.

All the three methods try to vacate room from the encrypted images directly. However, since the entropy of encrypted images has been maximized, these techniques can only achieve small payloads [16], [17] or generate marked image with poor quality for large payload [18] and all of them are subject to some error rates on data extraction and/or image restoration. Although the methods in [16], [17] can eliminate errors by error-correcting codes, the pure payloads will be further consumed.

In the present paper, we propose a novel method for RDH in encrypted images, for which we do not "vacate room after encryption" as done in [16]–[18], but "reserve room before encryption". In the proposed method, we first empty out room by embedding LSBs of some pixels into other pixels with a traditional RDH method and then encrypt the image, so the positions of these LSBs in the encrypted image can be used to embed data. Not only does the proposed method separate data extraction from image decryption but also achieves excellent performance in two different prospects:

- Real reversibility is realized, that is, data extraction and image recovery are free of any error.
- For given embedding rates, the PSNRs of decrypted image containing the embedded data are significantly improved; and for the acceptable PSNR, the range of embedding rates is greatly enlarged.

This paper is organized in the following manner. Section II briefly introduces previous methods proposed in [16]–[18]. The novel method is elaborated in Section III followed by some implementation issues in Section IV. Experiments with analysis and comparison are given in Section V. The paper is concluded in Section VI.

II. Previous Arts

The methods proposed in [16]–[18] can be summarized as the framework, "vacating room after encryption (VRAE)", as illustrated in Fig. 1(a).

In this framework, a content owner encrypts the original image using a standard cipher with an encryption key. After producing the encrypted image, the content owner hands over it to a data hider (e.g., a database manager) and the data hider can embed some auxiliary data into the encrypted image by losslessly vacating some room according to a data hiding key. Then a receiver, maybe the content owner himself or an authorized third party can extract the embedded data with the data hiding key and further recover the original image from the encrypted version according to the encryption key.

In all methods of [16]–[18], the encrypted 8-bit gray-scale images are generated by encrypting every bit-planes with a stream cipher. The method in [16] segments the encrypted image into a number of nonoverlapping blocks sized by $a \times a$; each block is used to carry one additional bit. To do this, pixels in each block are pseudo-randomly divided into two sets S_1 and S_2 according to a data hiding key. If the additional bit to be embedded is 0, flip the 3 LSBs of each encrypted pixel in S_1, otherwise flip the 3 encrypted LSBs of pixels in S_2. For data extraction and image recovery, the receiver flips all the three LSBs of pixels in S_1 to form a new decrypted block, and flips all the three LSBs of pixels in S_2 to form another new block; one of them will be decrypted to the original block. Due to spatial correlation in natural images, original block is presumed to be much smoother than interfered block and embedded bit can be extracted correspondingly. However, there is a risk of defeat of bit extraction and image recovery when divided block is relatively small (e.g., $a = 8$) or has much fine-detailed textures.

Hong *et al.* [17] reduced the error rate of Zhang's method [16] by fully exploiting the pixels in calculating the smoothness of each block and using side match. The extraction and recovery of blocks are performed according to the descending order of the absolute smoothness difference between two candidate blocks and recovered blocks can further be used to evaluate the smoothness of unrecovered blocks, which is referred to as side match.

Zhang's method in [18] pseudo-randomly permuted and divided encrypted image into a number of groups with size of L. The P LSB-planes of each group are compressed with a parity-check matrix and the vacated room is used to embed data. For instance, denote the pixels of one group by $x_1, \ldots, x_L$, and its encrypted P LSB-planes by $\mathbf{c}$ that consists of $P \cdot L$ bits. The data hider generates a parity-check matrix $\mathbf{G}$ sized $(P \cdot L - S) \times P \cdot L$, and compresses $\mathbf{c}$ as its syndrome $\mathbf{s}$ such that $\mathbf{s} = \mathbf{G} \cdot \mathbf{c}$. Because the length of $\mathbf{s}$ is $(P \cdot L - S)$, S bits are available for data accommodation. At the receiver side, the $8 - P$ most significant bits (MSB) of pixels are obtained by decryption directly. The receiver then estimates x_i $(1 \leq i \leq L)$ by the MSBs of neighboring pixels, and gets an estimated version of $\mathbf{c}$ denoted by $\mathbf{c}'$. On the other hand, the receiver tests each vector belonging to the coset $\Omega(\mathbf{s})$ of syndrome $\mathbf{s}$, where $\Omega(\mathbf{s}) = \{\mathbf{u} \mid \mathbf{G} \cdot \mathbf{u} = \mathbf{s}\}$. From each vector of $\Omega(\mathbf{s})$, the receiver can get a restored version of $\mathbf{c}$, and select the one most similar to the estimated version $\mathbf{c}'$ as the restored LSBs.

III. Proposed Method

Since losslessly vacating room from the encrypted images is relatively difficult and sometimes inefficient, why are we still so obsessed to find novel RDH techniques working directly for encrypted images? If we reverse the order of encryption and vacating room, i.e., reserving room prior to image encryption at content owner side, the RDH tasks in encrypted images would be more natural and much easier which leads us to the novel framework, "reserving room before encryption (RRBE)".

As shown in Fig. 1(b), the content owner first reserves enough space on original image and then converts the image into its encrypted version with the encryption key. Now, the data embed-

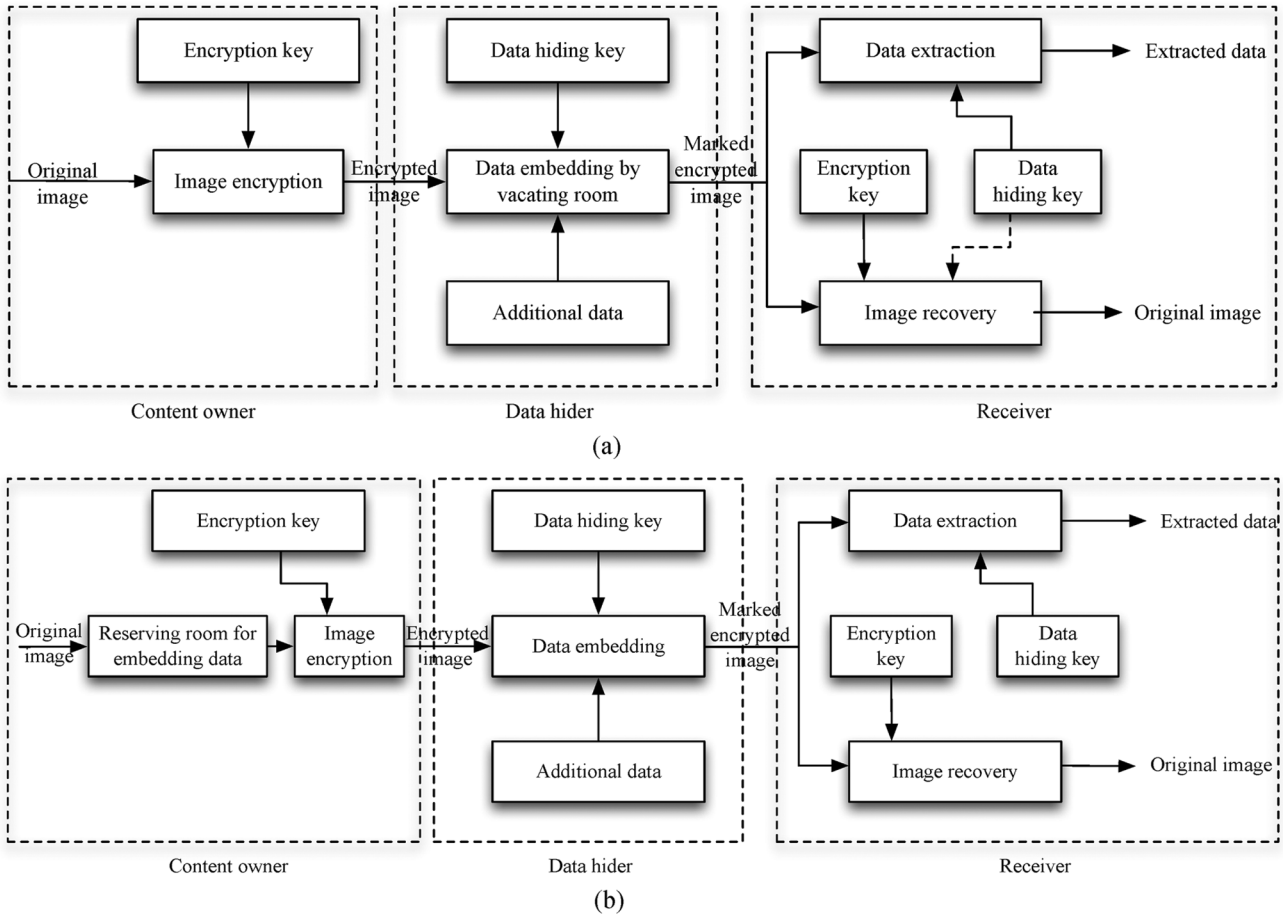


Fig. 1. Framework: "vacating room after encryption (VRAE)" versus framework: "reserving room before encryption (RRBE)." (Dashed line in (a) states that the need of data hiding key in image recovery varies in different practical methods). (a) Framework VRAE. (b) Framework RRBE.

ding process in encrypted images is inherently reversible for the data hider only needs to accommodate data into the spare space previous emptied out. The data extraction and image recovery are identical to that of Framework VRAE. Obviously, standard RDH algorithms are the ideal operator for reserving room before encryption and can be easily applied to Framework RRBE to achieve better performance compared with techniques from Framework VRAE. This is because in this new framework, we follow the customary idea that first losslessly compresses the redundant image content (e.g., using excellent RDH techniques) and then encrypts it with respect to protecting privacy.

Next, we elaborate a practical method based on the Framework "RRBE", which primarily consists of four stages: generation of encrypted image, data hiding in encrypted image, data extraction and image recovery. Note that the reserving operation we adopt in the proposed method is a traditional RDH approach.

A. Generation of Encrypted Image

Actually, to construct the encrypted image, the first stage can be divided into three steps: image partition, self reversible embedding followed by image encryption. At the beginning, image partition step divides original image into two parts $\mathbf{A}$ and $\mathbf{B}$; then, the LSBs of $\mathbf{A}$ are reversibly embedded into $\mathbf{B}$ with a standard RDH algorithm so that LSBs of $\mathbf{A}$ can be used for accommodating messages; at last, encrypt the rearranged image to generate its final version.

1) Image Partition: The operator here for reserving room before encryption is a standard RDH technique, so the goal of image partition is to construct a smoother area $\mathbf{B}$, on which standard RDH algorithms such as [10], [11] can achieve better performance. To do that, without loss of generality, assume the original image $\mathbf{C}$ is an 8 bits gray-scale image with its size $M \times N$ and pixels $\mathbf{C}_{i,j} \in [0, 255], 1 \le i \le M, 1 \le j \le N$. First, the content owner extracts from the original image, along the rows, several overlapping blocks whose number is determined by the size of to-be-embedded messages, denoted by l. In detail, every block consists of m rows, where $m = \lceil l/N \rceil$, and the number of blocks can be computed through $n = M - m + 1$. An important point here is that each block is overlapped by pervious and/or subsequential blocks along the rows. For each block, define a function to measure its first-order smoothness

$$f = \sum_{u=2}^{m} \sum_{v=2}^{N-1} \left| \mathbf{C}_{u,v} - \frac{\mathbf{C}_{u-1,v} + \mathbf{C}_{u+1,v} + \mathbf{C}_{u,v-1} + \mathbf{C}_{u,v+1}}{4} \right|. \tag{1}$$

Higher f relates to blocks which contain relatively more complex textures. The content owner, therefore, selects the particular block with the highest f to be $\mathbf{A}$, and puts it to the front of

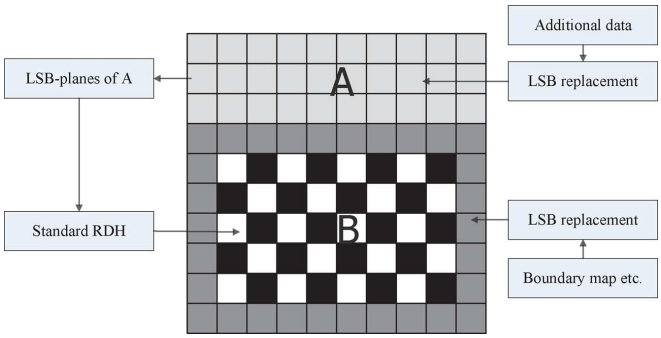


Fig. 2. Illustration of image partition and embedding process.

the image concatenated by the rest part $\mathbf{B}$ with fewer textured areas, as shown in Fig. 2.

The above discussion implicitly relies on the fact that only single LSB-plane of $\mathbf{A}$ is recorded. It is straightforward that the content owner can also embed two or more LSB-planes of $\mathbf{A}$ into $\mathbf{B}$, which leads to half, or more than half, reduction in size of $\mathbf{A}$. However, the performance of $\mathbf{A}$, in terms of PSNR, after data embedding in the second stage decreases significantly with growing bit-planes exploited. Therefore, in this paper, we investigate situations that at most three LSB-planes of $\mathbf{A}$ are employed and determine the number of bit-plane with regard to different payloads experimentally in the next section.

2) Self-Reversible Embedding: The goal of self-reversible embedding is to embed the LSB-planes of $\mathbf{A}$ into $\mathbf{B}$ by employing traditional RDH algorithms. For illustration, we simplify the method in [10] to demonstrate the process of self-embedding. Note that this step does not rely on any specific RDH algorithm.

Pixels in the rest of image $\mathbf{B}$ are first categorized into two sets: white pixels with its indices i and j satisfying $(i+j) \bmod 2 = 0$ and black pixels whose indices meet $(i+j) \bmod 2 = 1$, as shown in Fig. 2. Then, each white pixel, $\mathbf{B}_{i,j}$, is estimated by the interpolation value obtained with the four black pixels surrounding it as follows

$$\mathbf{B}'_{i,j} = w_1\mathbf{B}_{i-1,j} + w_2\mathbf{B}_{i+1,j} + w_3\mathbf{B}_{i,j-1} + w_4\mathbf{B}_{i,j+1}, \quad (2)$$

where the weight $w_i, 1 \le i \le 4$, is determined by the same method as proposed in [10]. The estimating error is calculated via $e_{i,j} = \mathbf{B}_{i,j} - \mathbf{B}'_{i,j}$ and then some data can be embedded into the estimating error sequence with histogram shift, which will be described later. After that, we further calculate the estimating errors of black pixels with the help of surrounding white pixels that may have been modified. Then another estimating error sequence is generated which can accommodate messages as well. Furthermore, we can also implement multilayer embedding scheme by considering the modified $\mathbf{B}$ as "original" one when needed. In summary, to exploit all pixels of $\mathbf{B}$, two estimating error sequences are constructed for embedding messages in every single-layer embedding process.

By bidirectional histogram shift, some messages can be embedded on each error sequence. That is, first divide the histogram of estimating errors into two parts, i.e., the left part and the right part, and search for the highest point in each part, denoted by LM and RM, respectively. For typical images, $LM = -1$ and $RM = 0$. Furthermore, search for the zero point in each part, denoted by LN and RN. To embed messages into positions with an estimating error that is equal to RM, shift all error values between $RM+1$ and $RN-1$ with one step toward right, and then, we can represent the bit 0 with RM and the bit 1 with $RM+1$. The embedding process in the left part is similar except that the shifting direction is left, and the shift is realized by subtracting 1 from the corresponding pixel values.

Suppose we should implement the embedding scheme x times to accommodate additional data. In the previous $x-1$ single-layer embedding rounds, peak points of two error sequences are selected and utilized to embed messages as above mentioned. When it comes to the xth single-layer embedding, only a small portion of messages is left to be embedded, so it is unadvisable to accommodate such little data at the expense of shifting all error values between peak points and their corresponding zero points. To deal with this issue, we can either exploit only part of error sequences which has enough peak points to embed the remaining messages while leaving the rest error sequences unchanged, or find two proper points, denoted by LP and RP, whose sum is larger, however closest to, the size of remaining messages. By shifting error values between LP, RP and their corresponding zero points, messages can be embedded into LP and RP instead of peak points. Fig. 3 illustrates the idea of selecting proper points. Generally speaking, two solutions can gain significantly improvement in terms of PSNR when the length of data is relatively short, i.e., when $x = 1$. And the superiority of one solution over the other depends highly on statistics of natural image itself which will be discussed in the next section.

The same with other RDH algorithms, overflow/underflow problem occurs when natural boundary pixels change from 255 to 256 or from 0 to -1. To avoid it, we only embed data into estimating error with its corresponding pixel valued from 1 to 254. However, ambiguities still arise when nonboundary pixels are changed from 1 to 0 or from 254 to 255 during the embedding process. These created boundary pixels in the embedding process are defined as pseudo-boundary pixels. Hence, a boundary map is introduced to tell whether boundary pixels in marked image are natural or pseudo in extracting process. It is a binary sequence with bit "0" for natural boundary pixel, bit "1" for pseudo-boundary pixel. Since estimating errors of marginal area of $\mathbf{B}$ cannot be calculated via (2), to make the best use of $\mathbf{B}$ we choose its marginal area shown in Fig. 2 to place the boundary map, and use LSB replacement to embed it. The original LSBs of marginal area is assembled with messages, i.e., LSB-planes of $\mathbf{A}$, and reversibly embedded into $\mathbf{B}$. In most cases, even with a large embedding rate, the length of boundary map is very short; thus, the marginal area of $\mathbf{B}$ is enough to accommodate it. Meanwhile, several parameters such as LN, RN, LM, RM, LP, RP, payloads embedded into the estimating errors of black pixels R_b, total embedding rounds x, start row SR and end row ER of $\mathbf{A}$ in original image, are embedded into marginal area in a similar way. These parameters play an important role in data extraction and image recovery process.

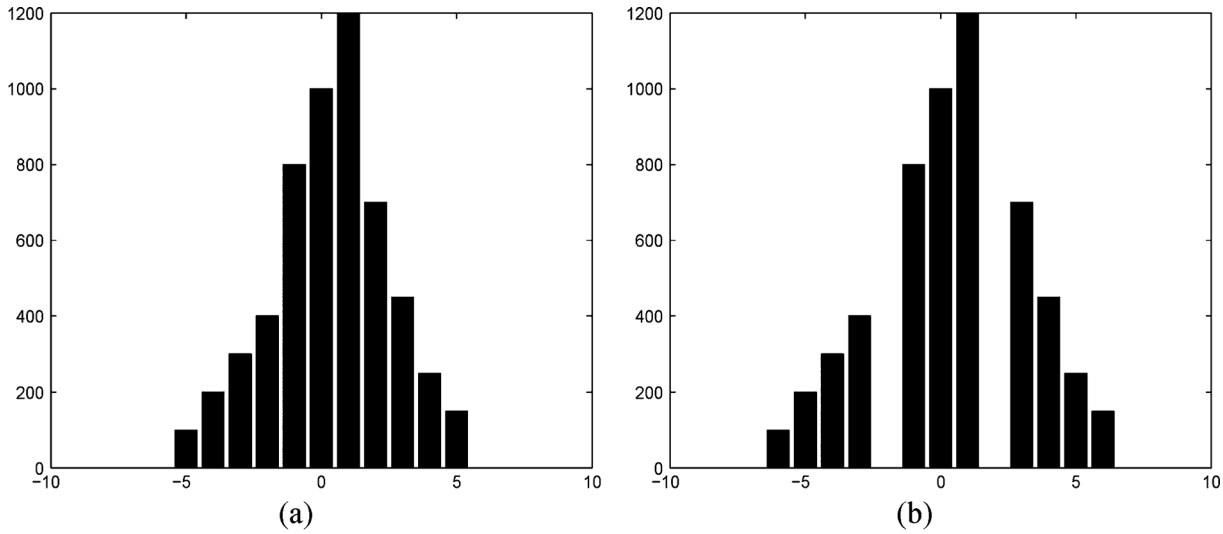


Fig. 3. Selection of proper points. (a) original histogram, (b) shifted histogram. (In this figure, length of messages is 1000 bits, $LP = -2$ and $RP = 2$.)

3) Image Encryption: After rearranged self-embedded image, denoted by $\mathbf{X}$, is generated, we can encrypts $\mathbf{X}$ to construct the encrypted image, denoted by $\mathbf{E}$. With a stream cipher, the encryption version of $\mathbf{X}$ is easily obtained. For example, a gray value $\mathbf{X}_{i,j}$ ranging from 0 to 255 can be represented by 8 bits, $\mathbf{X}_{i,j}(0), \mathbf{X}_{i,j}(1), \ldots, \mathbf{X}_{i,j}(7)$, such that

$$\mathbf{X}_{i,j}(k) = \left\lfloor \frac{\mathbf{X}_{i,j}}{2^k} \right\rfloor \bmod 2, \quad k = 0, 1, \ldots, 7. \tag{3}$$

The encrypted bits $\mathbf{E}_{i,j}(k)$ can be calculated through exclusive-or operation

$$\mathbf{E}_{i,j}(k) = \mathbf{X}_{i,j}(k) \oplus r_{i,j}(k), \tag{4}$$

where $r_{i,j}(k)$ is generated via a standard stream cipher determined by the encryption key. Finally, we embed 10 bits information into LSBs of first 10 pixels in encrypted version of $\mathbf{A}$ to tell data hider the number of rows and the number of bit-planes he can embed information into. Note that after image encryption, the data hider or a third party can not access the content of original image without the encryption key, thus privacy of the content owner being protected.

B. Data Hiding in Encrypted Image

Once the data hider acquires the encrypted image $\mathbf{E}$, he can embed some data into it, although he does not get access to the original image. The embedding process starts with locating the encrypted version of $\mathbf{A}$, denoted by $\mathbf{A}_E$. Since $\mathbf{A}_E$ has been rearranged to the top of $\mathbf{E}$, it is effortless for the data hider to read 10 bits information in LSBs of first 10 encrypted pixels. After knowing how many bit-planes and rows of pixels he can modify, the data hider simply adopts LSB replacement to substitute the available bit-planes with additional data $\mathbf{m}$. Finally, the data hider sets a label following $\mathbf{m}$ to point out the end position of embedding process and further encrypts $\mathbf{m}$ according to the data hiding key to formulate marked encrypted image denoted by $\mathbf{E}'$. Anyone who does not possess the data hiding key could not extract the additional data.

C. Data Extraction and Image Recovery

Since data extraction is completely independent from image decryption, the order of them implies two different practical applications.

1) Case 1: Extracting Data From Encrypted Images: To manage and update personal information of images which are encrypted for protecting clients' privacy, an inferior database manager may only get access to the data hiding key and have to manipulate data in encrypted domain. The order of data extraction before image decryption guarantees the feasibility of our work in this case.

When the database manager gets the data hiding key, he can decrypt the LSB-planes of $\mathbf{A}_E$ and extract the additional data $\mathbf{m}$ by directly reading the decrypted version. When requesting for updating information of encrypted images, the database manager, then, updates information through LSB replacement and encrypts updated information according to the data hiding key all over again. As the whole process is entirely operated on encrypted domain, it avoids the leakage of original content.

2) Case 2: Extracting Data From Decrypted Images: In Case 1, both embedding and extraction of the data are manipulated in encrypted domain. On the other hand, there is a different situation that the user wants to decrypt the image first and extracts the data from the decrypted image when it is needed. The following example is an application for such scenario. Assume Alice outsourced her images to a cloud server, and the images are encrypted to protect their contents. Into the encrypted images, the cloud server marks the images by embedding some notation, including the identity of the images' owner, the identity of the cloud server and time stamps, to manage the encrypted images. Note that the cloud server has no right to do any permanent damage to the images. Now an authorized user, Bob who has been shared the encryption key and the data hiding key, downloaded and decrypted the images. Bob hoped to get marked decrypted images, i.e., decrypted images still including the notation, which can be used to trace the source and history of the data. The order of image decryption before/without data extraction is perfectly suitable for this case. Next, we describe how to generate a marked decrypted image.

TABLE I
PSNR COMPARISON FOR THREE DIFFERENT LSB-PLANE CHOICES UNDER VARIOUS EMBEDDING RATES

		PSNR results (dB)							
embedding rate (bpp)		0.005	0.01	0.05	0.1	0.2	0.3	0.4	0.5
Lena	1 LSB-plane	67.16	63.44	55.46	52.33	49.07	45.00	40.65	35.84
	2 LSB-planes	66.48	62.65	54.69	51.55	48.39	45.10	42.56	39.46
	3 LSB-planes	64.41	60.94	52.95	49.96	46.79	43.98	41.91	39.53
Airplane	1 LSB-plane	65.94	63.18	57.02	54.20	50.98	48.26	44.67	40.78
	2 LSB-planes	65.48	62.33	55.91	53.05	49.87	48.10	45.05	42.73
	3 LSB-planes	63.47	60.97	53.87	50.79	47.65	45.79	43.88	42.19
Barbara	1 LSB-plane	65.39	62.56	55.56	51.46	47.68	43.56	39.24	34.80
	2 LSB-planes	65.00	61.85	54.72	50.71	47.25	43.70	40.78	37.53
	3 LSB-planes	63.33	60.50	53.16	49.36	45.98	42.81	40.34	37.58
Baboon	1 LSB-plane	57.49	55.71	50.19	46.17	40.68	35.87	31.16	25.92
	2 LSB-planes	57.43	55.47	49.87	45.92	40.41	36.47	33.08	29.85
	3 LSB-planes	57.10	55.13	49.23	45.40	40.09	36.33	32.96	30.19
Peppers	1 LSB-plane	63.77	61.30	54.17	51.02	46.00	42.08	36.91	——
	2 LSB-planes	63.67	60.53	53.50	50.50	46.16	42.65	39.47	35.76
	3 LSB-planes	62.34	59.54	52.22	49.18	45.43	42.10	39.40	36.87
Boat	1 LSB-plane	67.22	64.13	56.75	52.62	49.10	45.21	41.24	35.99
	2 LSB-planes	66.72	63.26	55.75	51.71	48.40	44.98	42.46	39.98
	3 LSB-planes	64.57	61.34	53.73	50.02	46.71	43.81	41.70	39.46

a) Generating the Marked Decrypted Image: To form the marked decrypted image $\mathbf{X}''$ which is made up of $\mathbf{A}''$ and $\mathbf{B}''$, the content owner should do following two steps.

- **Step 1**. With the encryption key, the content owner decrypts the image except the LSB-planes of $\mathbf{A}_E$. The decrypted version of $\mathbf{E}'$ containing the embedded data can be calculated by

$$\mathbf{X}''_{i,j}(k) = \mathbf{E}'_{i,j}(k) \oplus r_{i,j}(k) \tag{5}$$

and

$$\mathbf{X}''_{i,j} = \sum_{k=0}^{7} \mathbf{X}''_{i,j}(k) \times 2^k, \tag{6}$$

  where $\mathbf{E}'_{i,j}(k)$ and $\mathbf{X}''_{i,j}(k)$ are the binary bits of $\mathbf{E}'_{i,j}$ and $\mathbf{X}''_{i,j}$, obtained via (3) respectively.
- **Step 2**. Extract SR and ER in marginal area of $\mathbf{B}''$. By rearranging $\mathbf{A}''$ and $\mathbf{B}''$ to its original state, the plain image containing embedded data is obtained.

As can be seen, the marked decrypted image $\mathbf{X}''$ is identical to rearranged $\mathbf{X}$ except LSB-planes of $\mathbf{A}$. At the meantime, it keeps perceptual transparency compared with original image $\mathbf{C}$. More specifically, the distortion is introduced via two separate ways: the embedding process by modifying the LSB-planes of $\mathbf{A}$ and self-reversible embedding process by embedding LSB-planes of $\mathbf{A}$ into $\mathbf{B}$. The first part distortion is well controlled via exploiting the LSB-planes of $\mathbf{A}$ only and the second part can benefit from excellent performance of current RDH techniques.

b) Data Extraction and Image Restoration: After generating the marked decrypted image, the content owner can further extract the data and recover original image. The process is essentially similar to that of traditional RDH methods [10], [11]. The following outlines the specific steps:

- **Step 1**. Record and decrypt the LSB-planes of $\mathbf{A}''$ according to the data hiding key; extract the data until the end label is reached.
- **Step 2**. Extract $LN, RN, LM, RM, LP, RP, R_b, x$ and boundary map from the LSB of marginal area of $\mathbf{B}''$. Then, scan $\mathbf{B}''$ to undertake the following steps.
- **Step 3**. If R_b is equal to 0, which means no black pixels participate in embedding process, go to Step 5.
- **Step 4**. Calculate estimating errors $e'_{i,j}$ of the black pixels $\mathbf{B}''_{i,j}$. If $\mathbf{B}''_{i,j}$ belongs to [1, 254], recover the estimating error and original pixel value in a reverse order and extract embedded bits when $e'_{i,j}$ is equal to LN, LM(or LP), RM (or RP) and RN. Else, if $\mathbf{B}''_{i,j} \in \{0, 255\}$, refer to the corresponding bit b in boundary map. If $b = 0$, skip this one, else operate like $\mathbf{B}''_{i,j} \in [1, 254]$. Repeat this step until the part of payload R_b is extracted. If extracted bits are LSBs of pixels in marginal area, restore them immediately.
- **Step 5**. Calculate estimating errors $e'_{i,j}$ of the white pixels $\mathbf{B}''_{i,j}$, and extract embedded bits and recover white pixels in the same manner with Step 4. If extracted bits are LSBs of pixels in marginal area, restore them immediately.
- **Step 6**. Continue doing Step 2 to Step 5 $x - 1$ rounds on $\mathbf{B}''$ and merge all extracted bits to form LSB-planes of $\mathbf{A}$. Until now, we have perfectly recover $\mathbf{B}$.
- **Step 7**. Replace marked LSB-planes of $\mathbf{A}''$ with its original bits extracted from $\mathbf{B}''$ to get original cover image $\mathbf{C}$.

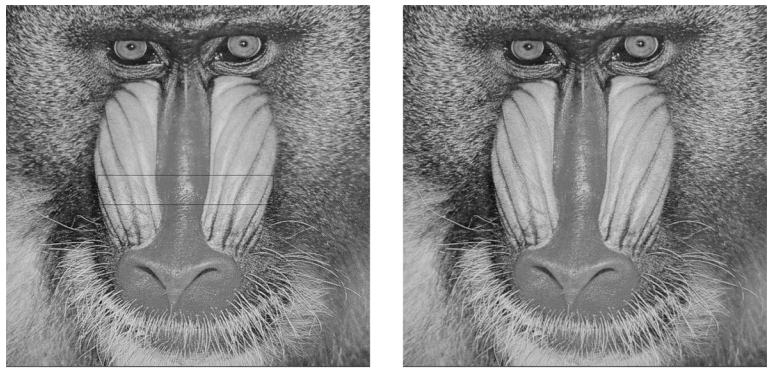
We note that if the content owner wants to retrieve his image in Case 1, the procedures are exactly the same in Case 2. Thus, it is omitted in Case 1 for simplicity.

IV. IMPLEMENTATION ISSUES

The proposed approach will be tested on public available standard images, which include "Lena", "Airplane", "Barbara", "Baboon", "Peppers" and "Boat" [19]. The size of all images is $512 \times 512 \times 8$. The objective criteria PSNR is employed to evaluate the quality of marked decrypted image quantitatively. To achieve high PSNRs, several implement details for the proposed method are discussed first.

A. Choice of LSB-Plane Number

When original image $\mathbf{C}$ is divided into $\mathbf{A}$ and $\mathbf{B}$, the size of $\mathbf{A}$ is determined not only by the length of to-be-embedded messages but also by the number of LSB-planes embedded reversibly in $\mathbf{B}$. The use of multiple LSB-planes takes into account the fact that the size of $\mathbf{B}$ can be enlarged with an in-



(a) (b)

Fig. 4. Emergence of "Cut" artifact of Baboon image (embedding rate is 0.5 bpp for visibility). (a) Single LSB-plane applied (rectangle area), (b) two LSB-planes applied.

TABLE II
EMBEDDING STRATEGIES ANALYSIS UNDER VARIOUS EMBEDDING RATES

		PSNR results (dB)							
embedding rate (bpp)		0.005	0.01	0.05	0.1	0.2	0.3	0.4	0.5
Lena	peak points	67.16	63.44	55.46	52.33	49.07	45.00	40.65	35.84
	proper points	64.53	62.05	55.90	51.64	48.99	44.83	40.54	36.08
Airplane	peak points	65.94	63.18	57.02	54.20	50.98	48.26	44.67	40.78
	proper points	63.89	62.74	57.46	53.98	51.09	48.48	44.91	40.52
Barbara	peak points	65.39	62.56	55.56	51.46	47.68	43.56	39.24	34.80
	proper points	59.62	58.08	53.63	51.04	47.10	43.02	39.24	34.88
Baboon	peak points	57.493	55.71	50.19	46.17	40.68	35.87	31.16	25.92
	proper points	59.61	56.80	50.49	46.26	40.51	35.91	31.07	25.94
Peppers	peak points	63.77	61.30	54.17	51.02	46.00	42.08	36.91	——
	proper points	64.71	62.31	51.20	51.23	46.11	42.20	37.10	——
Boat	peak points	67.22	64.13	56.75	52.62	49.10	45.21	41.24	35.99
	proper points	63.28	60.73	55.53	51.62	49.03	45.29	41.36	35.99

TABLE III
LENGTH OF BOUNDARY MAP UNDER DIFFERENT EMBEDDING RATES

	Boundary map size (bits)							
embedding rate (bpp)	0.005	0.01	0.05	0.1	0.2	0.3	0.4	0.5
Lena	0	0	0	0	0	0	0	0
Airplane	0	0	0	0	0	0	0	0
Barbara	0	0	0	0	0	0	0	0
Baboon	0	0	0	0	0	2	18	109
Peppers	0	1	43	92	291	797	1741	——
Boat	0	0	0	0	0	0	0	0

(a) (b) (c) (d)

Fig. 5. (a) Original image, (b) encrypted image, (c) decrypted image containing messages (embedding rate 0.1 bpp), (d) recovery version.

crease in embedding capability. Therefore, it is more likely that $\mathbf{B}$ only need to implement embedding scheme once to accommodate LSB-planes of $\mathbf{A}$, thus leading to distortion reduction. In other words, $\mathbf{A}$ shares part of distortion happens in $\mathbf{B}$. Table I shows the comparison results measured by PSNR for three different choices of LSB-planes (LSB-planes of $\mathbf{A}$ are embedded into peak points of estimating error sequences in $\mathbf{B}$), where the embedding rate is measured by bits per pixel (bpp). The choice of single LSB-plane outperforms the other two at low embedding rate levels (less than 0.2 bpp). It is consistent with our intuitive understanding: when embedding rate is small, $\mathbf{B}$ has the capacity to embed LSBs of $\mathbf{A}$ in a single round without size enlargement. Utilizing multiple LSB-planes can only introduce average distortion from 0.5 to 1.75 (case of two LSB-planes) in $\mathbf{A}$, calculated by mean squared error (MSE). With a growing embedding rate, the gain by choosing two LSB-planes is espe-

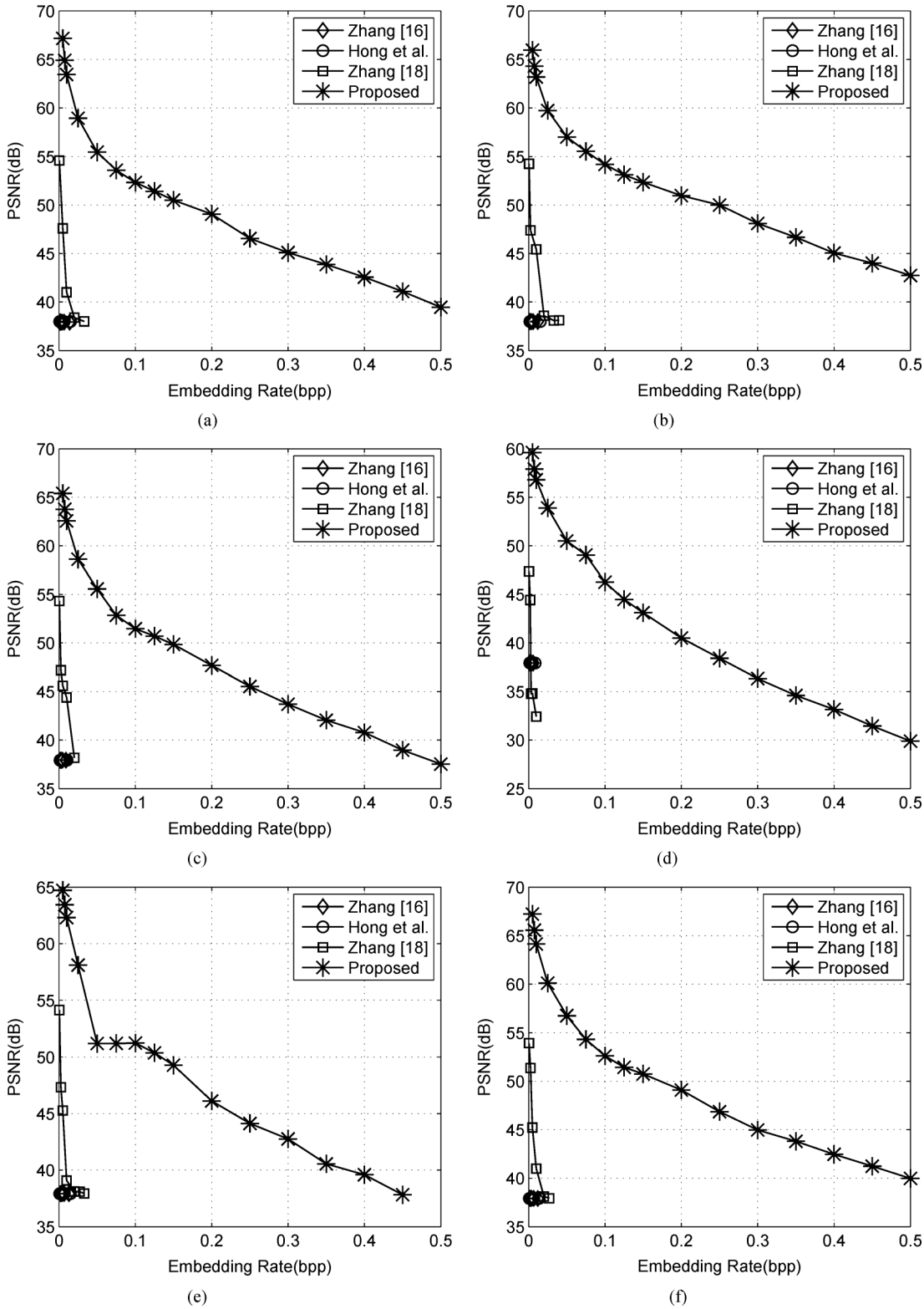


Fig. 6. PSNR comparison with the methods of Zhang [16], Hong [17] and Zhang [18]. (a) Lena, (b) Airplane, (c) Barbara, (d) Baboon, (e) Peppers, (f) Boat.

cially significant, where the improvement can be as high as 2 to 4 dB over selecting single LSB-plane.

Furthermore, we prefer using two LSB-planes to single one when their performance are competitive in embedding rate range from 0.2 to 0.3 bpp. This is because by allocating part distortion of $\mathbf{B}$ into $\mathbf{A}$, the "cut" artifact depicted in Fig. 4, can be reduced to a certain degree. Additionally, we cannot expect any significant improvement by exploiting three LSB-planes of $\mathbf{A}$ from the table. In practice, we utilize single LSB-plane to embed messages when embedding rate is less than 0.25 bpp, and switch to two LSB-planes with embedding rate larger than 0.25 bpp.

B. Choice of Embedding Strategy

In xth single-layer embedding we introduce two solutions for embedding only a small portion of messages: 1) embedding data into peak points by making use of part error sequence; and 2) searching for proper points in the histogram of all estimating errors. The comparison results are listed in Table II. The first solution performs better than the other when cover image is relatively smooth with little fine-detail regions, therefore resulting in a sharper representation in error histogram. The improvement can be as high as 2 to 4 dB at low embedding rate levels. As for textured images such as Baboon with rather flat error histogram, the second solution has a better performance of 1 to 2 dB. Note that the performance of two solutions gradually approaches the same with little difference at large embedding rate range. In this paper, we propose the first solution when peak points of estimating error sequence of cover image account for more than 20% of the whole errors; otherwise switch to the second.

C. Discussion on Boundary Map

Boundary map in this paper, is used for distinguishing between natural and pseudo boundary pixels and its size is critical to practical applicability of proposed approach. Table III shows the boundary map size of six standard images. In most cases, no boundary map is needed. Even for Peppers image, the largest size is 1741 bits (with a large embedding rate 0.4 bpp by adopting embedding scheme 4 rounds) and the marginal area ($512 \times 4 \times 4 = 8912$ bits) is large enough to accommodate it.

V. Experiments and Comparisons

We take standard image Lena, shown in Fig. 5(a), to demonstrate the feasibility of proposed method. Fig. 5(b) is the encrypted image containing embedded messages and the decrypted version with messages is illustrated in Fig. 5(c). Fig. 5(d) depicts the recovery version which is identical to original image.

We have compared the proposed method with the state-of-the-art works [16]–[18]. As mentioned in Section I, all methods in [16]–[18] maybe introduce some errors on data extraction and/or image restoration, while the proposed method is free of any error for all kinds of images.

The quality of marked decrypted images is compared in the term of PSNR. Fig. 6 plots the PSNR results of different marked decrypted images under given embedding rates. Out of fairness, we modify the methods in [16], [17] with error-correcting codes to eliminate errors. By introducing error-correcting codes, the pure payload of [16], [17] is reduced from *Cap* to $\mathrm{Cap}(1 - H(\rho))$, where $H(\rho)$ is the binary entropy function with error rate ρ. Take test image Baboon for instance. If each embedding block is sized of 8×8 with error rate 15.55% [16], then the pure payload is 1543 bits rather than 4096 bits. As for the method in [18], we only choose those results with a significantly high probability of successful data extraction and perfect image recovery to draw the curves. From the Fig. 6, it can be observed that over all range of embedding rate, for all cases, our approach outperforms state-of-the-art RDH algorithms in encrypted images. The gain in terms of PSNR is significantly high at embedding rate range that the methods in [16]–[18] can achieve. In addition, another advantage of our approach is the much wider range of embedding rate for acceptable PSNRs. In fact, the proposed method can embed more than 10 times as large payloads for the same acceptable PSNR (e.g., $\mathrm{PSNR} = 40$ dB) as the methods in [16]–[18], which implies a very good potential for practical applications.

VI. Conclusion

Reversible data hiding in encrypted images is a new topic drawing attention because of the privacy-preserving requirements from cloud data management. Previous methods implement RDH in encrypted images by vacating room after encryption, as opposed to which we proposed by reserving room before encryption. Thus the data hider can benefit from the extra space emptied out in previous stage to make data hiding process effortless. The proposed method can take advantage of all traditional RDH techniques for plain images and achieve excellent performance without loss of perfect secrecy. Furthermore, this novel method can achieve real reversibility, separate data extraction and greatly improvement on the quality of marked decrypted images.

References

[1] T. Kalker and F. M. Willems, "Capacity bounds and code constructions for reversible data-hiding," in *Proc. 14th Int. Conf. Digital Signal Processing (DSP2002)*, 2002, pp. 71–76.

[2] W. Zhang, B. Chen, and N. Yu, "Capacity-approaching codes for reversible data hiding," in *Proc 13th Information Hiding (IH'2011), LNCS 6958*, 2011, pp. 255–269, Springer-Verlag.

[3] W. Zhang, B. Chen, and N. Yu, "Improving various reversible data hiding schemes via optimal codes for binary covers," *IEEE Trans. Image Process.*, vol. 21, no. 6, pp. 2991–3003, Jun. 2012.

[4] J. Fridrich and M. Goljan, "Lossless data embedding for all image formats," in *Proc. SPIE Proc. Photonics West, Electronic Imaging, Security and Watermarking of Multimedia Contents*, San Jose, CA, USA, Jan. 2002, vol. 4675, pp. 572–583.

[5] J. Tian, "Reversible data embedding using a difference expansion," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 8, pp. 890–896, Aug. 2003.

[6] Z. Ni, Y. Shi, N. Ansari, and S. Wei, "Reversible data hiding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 3, pp. 354–362, Mar. 2006.

[7] D. M. Thodi and J. J. Rodriguez, "Expansion embedding techniques for reversible watermarking," *IEEE Trans. Image Process.*, vol. 16, no. 3, pp. 721–730, Mar. 2007.

[8] X. L. Li, B. Yang, and T. Y. Zeng, "Efficient reversible watermarking based on adaptive prediction-error expansion and pixel selection," *IEEE Trans. Image Process.*, vol. 20, no. 12, pp. 3524–3533, Dec. 2011.

[9] P. Tsai, Y. C. Hu, and H. L. Yeh, "Reversible image hiding scheme using predictive coding and histogram shifting," *Signal Process.*, vol. 89, pp. 1129–1143, 2009.

[10] L. Luo *et al.*, "Reversible image watermarking using interpolation technique," *IEEE Trans. Inf. Forensics Security*, vol. 5, no. 1, pp. 187–193, Mar. 2010.

[11] V. Sachnev, H. J. Kim, J. Nam, S. Suresh, and Y.-Q. Shi, "Reversible watermarking algorithm using sorting and prediction," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 7, pp. 989–999, Jul. 2009.

[12] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL, USA: CRC, 1996.

[13] K. Hwang and D. Li, "Trusted cloud computing with secure resources and data coloring," *IEEE Internet Comput.*, vol. 14, no. 5, pp. 14–22, Sep./Oct. 2010.

[14] M. Johnson, P. Ishwar, V. M. Prabhakaran, D. Schonberg, and K. Ramchandran, "On compressing encrypted data," *IEEE Trans. Signal Process.*, vol. 52, no. 10, pp. 2992–3006, Oct. 2004.

[15] W. Liu, W. Zeng, L. Dong, and Q. Yao, "Efficient compression of encrypted grayscale images," *IEEE Trans. Image Process.*, vol. 19, no. 4, pp. 1097–1102, Apr. 2010.
[16] X. Zhang, "Reversible data hiding in encrypted images," *IEEE Signal Process. Lett.*, vol. 18, no. 4, pp. 255–258, Apr. 2011.
[17] W. Hong, T. Chen, and H. Wu, "An improved reversible data hiding in encrypted images using side match," *IEEE Signal Process. Lett.*, vol. 19, no. 4, pp. 199–202, Apr. 2012.
[18] X. Zhang, "Separable reversible data hiding in encrypted image," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 2, pp. 826–832, Apr. 2012.
[19] Miscelaneous Gray Level Images [Online]. Available: http://decsai.ugr.es/cvg/dbimagenes/g512.php

Kede Ma received the B.S. degree in electronic engineering and information science from the University of Science and Technology of China. He is currently working toward the Master degree at the University of Waterloo. His research interests include information hiding and image quality assessment.

Weiming Zhang received the M.S. and Ph.D. degrees in 2002 and 2005, respectively, from the Zhengzhou Information Science and Technology Institute, China.

Currently, he is an associate professor with the School of Information Science and Technology, University of Science and Technology of China. His research interests include information hiding and multimedia security.

Xianfeng Zhao (S'03–A'04–M'04) received the Ph.D. degree in computer science from Shanghai Jiao Tong University, Shanghai, China, in 2003.

From 2003 to 2005, he was a postdoctoral fellow with the Data Assurance and Communication Security Center, Chinese Academy of Sciences (CAS), Beijing. From 2006 to 2011, he was an associate professor with the State Key Laboratory of Information Security (SKLOIS), Institute of Software, CAS, Beijing. Since 2012, he has been a professor with SKLOIS, which was moved to Institute of Information Engineering, CAS, Beijing, in 2012. He is a member of the China Computer Federation and the Chinese Association for Cryptologic Research. His research interests include information hiding and multimedia security.

Nenghai Yu received the B.S. degree in 1987 from Nanjing University of Posts and Telecommunications, the M.E. degree in 1992 from Tsinghua University, and the Ph.D. degree in 2004 from the University of Science and Technology of China, where he is currently a professor. His research interests include multimedia security, multimedia information retrieval, video processing, and information hiding.

Fenghua Li received the B.S., M.S., and Ph.D. degrees in computer software and computer systems architecture from Xidian University, China, in 1987, 1990, and 2009, respectively.

He was a lecturer with Xidian University from 1992 to 1994. He became an associate professor in 1995 and a professor in 2001 with Beijing Electronic Science and Technology Institute. Since 2011, he has been with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences as a professor, and doctoral supervisor director. His research interests include network security, system security and evaluation, and trusted computation.