Signal Processing: Image Communication 64 (2018) 78–88

Contents lists available at ScienceDirect

Signal Processing: Image Communication

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

Reversible data hiding in encrypted images using adaptive block-level prediction-error expansion

Shuang Yi [a], Yicong Zhou [a,*], Zhongyun Hua [b]

[a] Department of Computer and Information Science, University of Macau, Macau 999078, China
[b] School of Computer Science and Technology, Harbin Institute of Technology Shenzhen Graduate School, Shenzhen 518055, China

ARTICLE INFO

Keywords:
Reversible data hiding
Encrypted images
Adaptive embedding
Block-level prediction-error expansion

ABSTRACT

As directly reserving room from the encrypted image for data embedding is difficult and inefficient, many encryption domain based reversible data hiding schemes have disadvantages such as small embedding rate and low visual quality of the directly decrypted image. In order to solve these problems, this paper first introduces a reversible data hiding method for natural images using the block-level prediction-error expansion. The method can embed secret data into 2×2 image blocks by exploiting the pixel redundancy within each block. Extending this concept to the encrypted domain, we then propose a reversible data hiding method in encrypted images using adaptive block-level prediction-error expansion (ABPEE-RDHEI). ABPEE-RDHEI encrypts the original image by block permutation to preserve spatial redundancy for data embedding, and applies a stream cipher to the block permutated image to further enhance the security level. Due to the adaptive pixel selection and iterative embedding processes, the proposed ABPEE-RDHEI can achieve a high embedding rate and pleasing visual quality of the marked decrypted images. Experimental results and analysis show that ABPEE-RDHEI has a better performance than several state-of-the-art methods.

1. Introduction

Reversible data hiding (RDH) in images aims to embed the secret data into an original image in an imperceptible way. Unlike watermarking that should be robust to against malicious attacks, RDH emphasizes perfect data extraction and image recovery at the receiver side. Many efficient RDH methods have been proposed in recent years. The early RDH is based on image compression, where a number of possible features of the original image are extracted and losslessly compressed. The reserved spare space is utilized for embedding the secret data. Later, more RDH methods have been proposed by exploiting the pixel spatial correlations of the image (e.g., difference expansion (DE) [1] and histogram shifting (HS) [2–4]) and exploiting modification direction (EMD) [5,6]. The DE methods embed secret data by enlarging the difference of two adjacent pixel values, and HS based methods embed secret data into the shifted histograms. Another commonly used RDH method is called prediction-error expansion (PEE) [7,8]. It exploits the prediction-error to embed the secret data using the HS technique.

Nowadays, many researchers show their interest in developing reversible data hiding methods in encrypted images (RDHEI), where the original image is first encrypted by the content provider, and then the data hider embeds secret data into the encrypted image without knowing the original image content. At the receiver side, RDHEI aims to completely recover both the secret data and original image. With the reversibility property, RDHEI has wide applications in many critical scenarios such as medical image sharing, law forensics, Cloud storage and military applications. If users want to store their images into the Cloud but do not want any unauthorized access, they can encrypt the images before sending them to the Cloud. Although the Cloud does not know the image contents, it is able to add some additional information to the encrypted images for the management of the resources (e.g., add notations or location information to the encrypted images). In this scenario, no transmission is involved, and thus no errors or attacks either [9].

Existing RDHEI methods can be divided into two categories: one is called vacating room after encryption (VRAE), the other is reserving room before encryption (RRBE) [10]. In the VRAE methods, the content provider needs do nothing but encrypt the original image. Many VRAE methods have been proposed in recent years [11–19]. They use the stream cipher [12–15,20–22], permutation [23–26] or Paillier cryptosystem [27] to encrypt the original image, and embed secret data by bits flipping [20–22], compression [28,29], HS [23–26,30] et al. However, vacating room for data embedding after image encryption

* Corresponding author.
E-mail address: yicongzhou@umac.mo (Y. Zhou).

https://doi.org/10.1016/j.image.2018.03.001
Received 3 November 2017; Received in revised form 26 January 2018; Accepted 2 March 2018
Available online 8 March 2018

may be difficult and inefficient. In order to increase the embedding rate, Ma et al. [10] proposed a RRBE method to reserve room from the original image before image encryption. Secret data then can be embedded into the reserved spare space directly. Later, some RRBE methods have been proposed by reserving the spare space using different techniques such as traditional RDH methods [10,31–33] and sparse representation technique [34].

Although RRBE can achieve a relatively high embedding rate, the content owner is required to perform an extra operation of reserving a spare space before encrypting the original image. The content owner may have difficulty to accomplish this extra operation to vacate rooms and/or may have no idea about the forthcoming secret data to be embedded. Therefore, many researchers show interest in developing VRAE methods. Existing VRAE methods for RDHEI can be divided into two categories, namely separable VRAE (S-VRAE) and joint (J-VRAE) methods. The former one can perform data extraction and image recovery separately, while the latter one cannot. Due to the difficulty of vacating room from the encrypted image, the VRAE methods are limited in the embedding rate. In addition, most VRAE methods extract the secret data and recover the original image by utilizing the fluctuation measure function to evaluate the smoothness of the decrypted image. They may fail to obtain the precise measure results when the images contain many textures. This leads to incorrect data extraction and/or partial image recovery. To address these problems, this paper proposes an adaptive block-level prediction-error expansion (ABPEE) to perform RDHEI. The main contributions of this work are summarized as follows:

(1) We propose a new block-level predictor (BLP) to predict the pixel values within a 2×2 image block. It can obtain more precise prediction results than the commonly used predictor like median edge detector (MED) [35].
(2) Using BLP, we introduce a block-level prediction-error expansion (BPEE) method to embed secret data into an image. It embeds secret data into image block by block rather than the conventional PEE that embeds data pixel by pixel in the raster-scan order.
(3) We further propose an Adaptive BPEE based RDHEI (ABPEE-RDHEI) scheme.
   (a) It inherits the merits of VRAE that reserving room process is not required at the content-owner side. In addition, data extraction and image recovery can be performed separately and independently.
   (b) It is fully reversible. This means that the secret data and original image can be recovered without any error.
   (c) It can achieve a high embedding rate, as iterative embedding is used.
   (d) It can obtain a larger embedding rate and generate marked decrypted images with higher visual quality than several state-of-the-art methods.

The rest of this paper is organized as follows: Section 2 briefly reviews some related works. Section 3 introduces BPEE. Section 4 presents the proposed ABPEE-RDHEI. Section 5 discusses several characteristics of ABPEE-RDHEI. Section 6 provides simulation results and comparisons with the state-of-the-art methods. Finally, Section 7 concludes this paper.

2. Related works

In this section, we review some commonly used predictors in image processing and some existing VRAE methods for RDHEI.

2.1. Existing predictors

Recently, many predictors have been proposed to estimate pixel values in spatial domain, such as MED [35], gradient adjusted predictor (GAP) [36], simplified gradient adjusted predictor (SGAP) [37], partial differential equations (PDE) predictor [38] and checkerboard based prediction (CBP) [39]. GAP and SGAP use the pixels in a half-surrounded structure with an irregular shape to predict the target pixel. They are suitable for predicting pixels in a raster-scan order. PDE completes the prediction process using the predefined reference pixels. CBP uses 25% pixels in a host image to predict the remaining 75% pixels, and each target pixel is predicted by pixels with a 3×3 region. MED uses pixels within a 2×2 image block to predict the target pixel, and it is described as follows:

$$\hat{x} = \begin{cases} \min(x_r, x_c), & \text{if } x_d \geqslant \max(x_r, x_c) \\ \max(x_r, x_c), & \text{if } x_d \leqslant \min(x_r, x_c) \\ x_r + x_c - x_d, & \text{otherwise} \end{cases} \quad (1)$$

where (x, x_r, x_c, x_d) denotes the pixels within a 2×2 image block, x is the target pixel, x_r, x_c and x_d are the three remaining pixels located in the same row, column, and diagonal directions with x, respectively.

2.2. VRAE methods

Here, we review two types of VRAE methods, J-VRAE and S-VRAE, separately.

2.2.1. J-VRAE methods

The J-VRAE methods usually first encrypt the original image using a stream cipher, and then embed the secret data using different techniques such as least significant bits (LSBs) flipping [12–15,20–22] and public key modulation [16]. Zhang's method [20] divides the encrypted image into a number of non-overlapped blocks, separates pixels in each block into two groups, namely S_0 and S_1, and embeds one bit of secret data into one image block by flipping the 3 LSBs of S_0 (if secret data bit is 0) or S_1 (if secret data bit is 1). Data extraction and image recovery are accomplished by comparing the smoothness of the decrypted image blocks. Yu et al. [12] proposed an improved version of Zhang's method [20] by randomly selecting p% ($p \in (0, 100]$) of pixels in the encrypted image as the active pixels, and flipping the 3 LSBs of all active pixels in S_0 or S_1 of an image block to embed different values of secret data bits. Thus, Zhang's method [20] is a special case of Yu et al.'s method [12] with $p = 100$. Hong et al. [21] and Liao et al.'s [22] methods use a more precise fluctuation measure function to calculate the complexity of the image block, and apply the side match technique to reduce the rate of incorrect data extraction. Wu et al. [13] embed one bit of the secret data into a group of pixels by flipping their ith($1 \leqslant i \leqslant 6$) least significant bits (LSBs). In Li et al.'s method [14], one bit of the secret data is embedded into a pre-selected pixel by flipping its 3 LSBs. In addition, it duplicates the secret data before embedding to reduce the rate of incorrect data extraction. Method in [15] flips only the LSBs of fewer pixels by the elaborate selection, so that visual quality of the marked decrypted image can be improved. Meanwhile, the adaptive judging function based on the distribution characteristics of image local contents effectively decreases the error rate of extracted secret data bits. Zhou et al. [16] use a public key modulation method to embed n ($n \geqslant 1$) bits of secret data into an image block. The support vector machine technique is utilized for data extraction and image recovery.

2.2.2. S-VRAE methods

In order to perform data extraction and image recovery separately, a number of S-VRAE methods have been proposed. Wu et al. [13] embed the secret data by replacing the ith($i \geqslant 7$) bit of a stream-cipher-encrypted pixel. Methods in [28,29] and [18] encrypt the original image using the stream cipher. In [29] and [28], a number of LSB planes and the 4th bit plane of a stream-cipher-encrypted image are compressed to accommodate secret data, respectively. Zhang et al. [18] use the pseudo random sequence modulation technique to embed secret data into 3 LSB planes of the encrypted image. Previous S-VRAE methods [11,13,18,28,29] can perfectly extract the secret data without any error but the recovered image may have data loss. This is because image recovery is accomplished by analyzing the local standard

deviation or the smoothness of the decrypted image. This may result in incorrect results of image recovery when the original image contains many textures. Different from these S-VRAE methods that encrypt the original image mainly using the stream cipher with the pixel locations unchanged, S-VRAE methods in [23–26] encrypt the original image by permutation. In [24], the original image is first decomposed by integer discrete wavelet transform (DWT) to obtain four frequency subbands (LL, HL, HL, HH) of coefficients, the secret data is then hidden into the Arnold map permutated HL, LH and HH subbands using HS method. Yi et al. [25] improved [24] by using 1/4 of the pixels in an original image to predict the rest 3/4 pixels, and the secret data is embedded into the permutated prediction-error values by HS. In [26], the original image is first divided into a number of blocks, and a coarse-grained permutation is utilized to permutate blocks in the whole image and a fine-grained permutation is applied to permutate each pixel within the block. Two pixels in each block are then randomly selected to be the peak points to embed secret data by HS. It is lossless in both secret data extraction and image recovery. Methods in [40] and [30] divide the original image into blocks and uses stream encryption and permutation to encrypt each block. Then, [40] uses two traditional RDH methods, difference histogram shifting (DHS) and prediction-error histogram shifting (PEHS), to embed secret data into the encrypted image, and [30] embeds secret data into the l LSBs of each block by HS with two randomly selected peak points. Niu et al. [41] encrypt the AMBTC compressed image and use HS to embed secret data into the encrypted lower mean value of each block. Method in [19] obtains the encrypted image by encrypting the integer wavelet transformed coefficients. Secret data is then embedded into the encrypted coefficients via the HS method. However, the embedding rate is still limited.

3. Block-level prediction-error expansion

3.1. Block-level predictor (BLP)

Here, we propose a BLP to predict the target pixel x within a 2×2 image block by considering its 3 remaining pixels with different weights. It is described as

$$\hat{x} = \lfloor w_r x_r + w_c x_c + w_d x_d \rfloor \tag{2}$$

where (x, x_r, x_c, x_d) denotes the pixels within a 2×2 image block, x is the target pixel, x_r, x_c and x_d are the three remaining pixels located in the same row, column, and diagonal directions with x, respectively. $\lfloor \gamma \rfloor$ is a floor function to obtain the largest integer not greater than γ, weight coefficients w_r, w_c and w_d are satisfied with $0 \leqslant w_r, w_c, w_d \leqslant 1$, $w_r + w_c + w_d = 1$. For simplicity, in this work, we set $w_r = w_c = 0.4$ and $w_d = 0.2$.

3.2. Block-level prediction-error expansion (BPEE)

The original image with size of $n_1 \times n_2$ is first divided into 2×2 non-overlapped blocks. Thus, totally $N = n_1 n_2/4$ blocks will be obtained. The locations of four pixels within a block are shown in Fig. 1(a).

In data embedding process, for each block, we first use the remaining three pixels x^2, x^3 and x^4 to predict x^1 using BLP, the obtained prediction-error values are utilized for secret data embedding. After the first round of embedding, pixel x^1 in each block is changed into $\tilde{x}^1$. The modified pixel $\tilde{x}^1$ together with the remaining pixels x^3 and x^4 will be utilized to predict pixel x^2 in the second round of embedding and it will change x^2 into $\tilde{x}^2$. Similarly, we can obtain $\tilde{x}^3$ and $\tilde{x}^4$.

After four rounds of data embedding, all four pixels in each image block are modified, as shown in Fig. 1(e). This whole process is called an embedding layer. After the first embedding layer, image pixels in all blocks are changed and thus a new image can be obtained to do the next embedding layer. These iterative embedding processes stop once all payload bits are completely embedded.

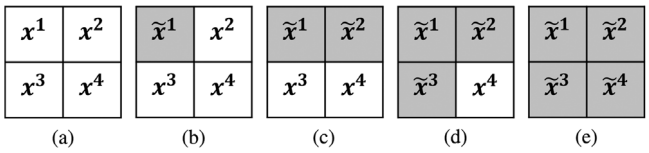


Fig. 1. Pixel changes in each embedding round. (a) The original pixels; pixel changes after the (b) first; (c) second; (d) third and (e) fourth embedding rounds, respectively.

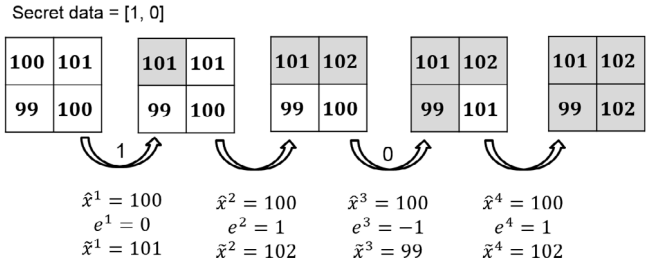


Fig. 2. Example of data embedding in each round of a single embedding layer when $T_l^k = -1$ and $T_r^k = 0$.

Without loss of generality, we use a single round of embedding to describe the data embedding procedure of BPEE. Suppose that we embed additional data into the kth ($1 \leqslant k \leqslant 4$) pixel of each block. Denote the kth pixel in the ith block as x_i^k, where $1 \leqslant i \leqslant N$, and N is the total number of blocks in the image. Firstly, we concatenate x_i^k and form a pixel sequence $(x_1^k, x_2^k, \ldots, x_N^k)$. Then the prediction value $\hat{x}_i^k$ of x_i^k is calculated by BLP using Eq. (2), and the obtained prediction-error sequence is denoted as $\mathbf{E}^k = (e_1^k, e_2^k, \ldots, e_N^k)$, where

$$e_i^k = x_i^k - \hat{x}_i^k \tag{3}$$

Based on $\mathbf{E}^k$, two capacity parameters T_l^k and T_r^k are obtained by

$$\begin{cases} T_l^k = \min\{\arg\max\limits_{e<0}\{h_{\mathbf{E}^k}(e)\}\} \\ T_r^k = \max\{\arg\max\limits_{e\geqslant 0}\{h_{\mathbf{E}^k}(e)\}\} \end{cases} \tag{4}$$

where $h_{\mathbf{A}}(t)$ is the number of occurrence when prediction-error values in the sequence $\mathbf{A}$ are equal to t. These two capacity parameters are then expanded to embed additional data bits by

$$\hat{e}_i^k = \begin{cases} e_i^k - 1, & \text{if } e_i^k < T_l^k \\ e_i^k - m, & \text{if } e_i^k = T_l^k \\ e_i^k + m, & \text{if } e_i^k = T_r^k \\ e_i^k + 1, & \text{if } e_i^k > T_r^k \\ e_i^k, & \text{otherwise} \end{cases} \tag{5}$$

where $m \in \{0, 1\}$ is one bit of the additional data. Therefore, one embedding round can embed $(h_{\mathbf{E}^k}(T_l^k) + h_{\mathbf{E}^k}(T_r^k))$ bits of the additional data. Finally, we generate new values of these pixels by

$$\tilde{x}_i^k = \hat{e}_i^k + \hat{x}_i^k \tag{6}$$

Fig. 2 shows an example of data embedding in each round of a single embedding layer. We assume that $T_l^k = -1$, $T_r^k = 0$, and secret data $= [1, 0]$. In the first round, $e^1 = T_r^k = 0$, the secret data bit '1' is embedded into the first pixel. In the third round, $e^3 = T_l^k = -1$, the secret data bit '0' is embedded into the third pixel. After four rounds of embedding, the original image block $B = [100, 101; 99, 100]$ becomes to $\tilde{B} = [101, 102; 99, 102]$. Then, we can regard $\tilde{B}$ as a new original block to do the second layer of embedding following the same procedures in Fig. 2.

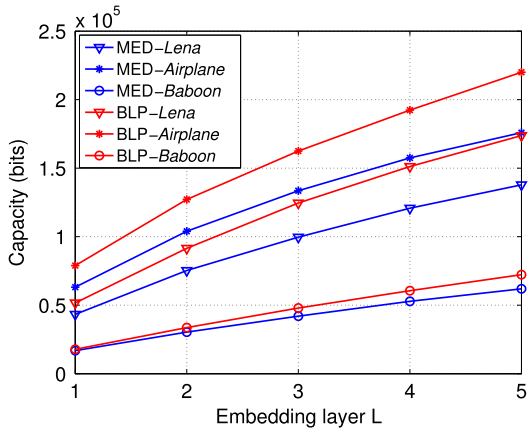


Fig. 3. Comparison of embedding capacity of BPEE using BLP and MED for different images and embedding layers.

3.3. Data extraction

The data extraction procedures follow the inverse order of embedding steps starting from pixel $\tilde{x}^4$ to $\tilde{x}^1$, as shown in Fig. 1 from (e) to (a). Thus, in each data extraction round, we can obtain the same prediction-error values as in the data embedding phase.

For each block, we first use the remaining three pixels $\tilde{x}^1$, $\tilde{x}^2$ and $\tilde{x}^3$ to predict $\tilde{x}^4$ using BLP, and the obtained prediction-error values are utilized for data extraction. After the first round of data extraction, pixel $\tilde{x}^4$ is changed as x^4, as shown in Fig. 1(d). Then, x^4 together with $\tilde{x}^1$ and $\tilde{x}^2$ are utilized to predict $\tilde{x}^3$, and after data extraction, $\tilde{x}^3$ is changed as x^3. Similarly, we can obtain x^1 and x^2.

After four rounds of data extraction, all four pixels in each image block are recovered. We use these recovered pixels to perform the next layer of extraction until all data bits are successfully extracted.

Here, we use a single round of extraction as an example to describe the detailed data extraction procedures of BPEE. We first denote the pixel sequence containing of the kth pixel in each block as $(\tilde{x}_1^k, \tilde{x}_2^k, \ldots, \tilde{x}_N^k)$, and then calculate its prediction values $(\hat{x}_1^k, \hat{x}_2^k, \ldots, \hat{x}_N^k)$ using BLP. The obtained prediction-error values are calculated by

$$\hat{e}_i^k = \tilde{x}_i^k - \hat{x}_i^k \tag{7}$$

According to the capacity parameters T_l^k and T_r^k, secret data can be extracted by

$$m = \begin{cases} 0, & \text{if } \hat{e}_i^k \in \{T_l^k, T_r^k\} \\ 1, & \text{if } \hat{e}_i^k \in \{T_l^k - 1, T_r^k + 1\} \end{cases} \tag{8}$$

and the prediction-error values are recovered by

$$e_i^k = \begin{cases} \hat{e}_i^k + 1, & \text{if } \hat{e}_i^k < T_l^k \\ \hat{e}_i^k - 1, & \text{if } \hat{e}_i^k > T_r^k \\ \hat{e}_i^k, & \text{otherwise} \end{cases} \tag{9}$$

Finally, the pixel values are reconstructed by

$$x_i^k = \hat{x}_i^k + e_i^k \tag{10}$$

3.4. Comparison

Here, we compare the pixel prediction performance of our proposed BLP with existing MED using 3 test images from the database.[1] Fig. 3 plots the embedding capacities of BPEE using BLP and MED for different embedding layers. We can observe that, the embedding capacity of BPEE using both predictors increases with the increasing of the embedding layer. However, BPEE using BLP can embed more payload bits than that using MED under various embedding layers. This means that BLP can obtain more precise prediction results than MED.

[1] http://decsai.ugr.es/cvg/dbimagenes/g512.php.

4. ABPEE-RDHEI

Using the proposed BLP and BPEE, this section introduces an adaptive block-level based prediction-error expansion for RDHEI (ABPEE-RDHEI). The ABPEE-RDHEI framework is shown in Fig. 4. The image provider first encrypts the original image using a block-level permutation with the permutation key $\mathcal{K}_p$. Then, using the sharing key $\mathcal{K}_s$, a stream encipher is utilized to change the pixel values of the image. At the data hider side, the stream decipher is applied using $\mathcal{K}_s$. According to the data embedding key $\mathcal{K}_d$, the data hider then embeds secret data into the encrypted image using the ABPEE method. After that, the stream encipher is used to further protect the secret data and image content. Note that the stream encipher and decipher can be replaced by any secure encryption algorithm, and in this paper we use stream cipher for demonstration. The encrypted image containing secret data is called the marked encrypted image. At the receiver side, we propose two schemes to extract the secret data and recover the original image to fulfil different applications. One scheme is data extraction before image decryption while the other is data extraction after image decryption. Using the image encryption key (or data embedding key), the receiver can directly obtain the marked decrypted image (or secret data). With both two security keys, the receiver can completely recover the original image and the secret data simultaneously.

4.1. Image encryption

The image encryption consists of two steps: block permutation and stream encipher. The block permutation contains a coarse-grained and a fine-grained permutation. Assume that the size of a gray-scale image $\mathbb{I}$ is $N_1 \times N_2$ and its pixel values are in the range of $[0, 255]$, where N_1 and N_2 are multiples of 2. We first divide $\mathbb{I}$ into a series of 2×2 non-overlapped blocks. Thus, $N = N_1 N_2/4$ blocks are obtained. According to $\mathcal{K}_p$, two permutations are applied to $\mathbb{I}$ to obtain the image $\hat{\mathbb{I}}$: a coarse-grained permutation to permutate blocks within the image and a fine-grained permutation to permutate pixels within each block. Note that, the fine-grained permutation is to rotate a 2×2 block in one of the four degrees clockwise: 0, 90, 180 and 270. Thus, there are totally $N!4^N$ possible permutations. Then, a stream encipher is applied to $\hat{\mathbb{I}}$ using key $\mathcal{K}_s$ to obtain the encrypted image $\mathbb{E}$. We can simply use $\mathcal{K}_s$ to generate a bit sequence and do the bit-XOR with $\hat{\mathbb{I}}$ to obtain $\mathbb{E}$.

4.2. Data embedding

After obtaining the encrypted image $\mathbb{E}$, the data hider first performs the stream decipher using $\mathcal{K}_s$. Here the same bit sequence is generated by $\mathcal{K}_s$ and then the bit-XOR is performed on $\mathbb{E}$ to obtain the image $\hat{\mathbb{I}}$. Because the block permutation in image encryption does not change the values and relative locations of pixels within a 2×2 image block, these pixels have strong spatial correlations similar to the original image, and their values are close to each other. Thus, we can embed secret data bits into these image blocks using the pixel spatial correlations and $\mathcal{K}_d$. After data embedding, the stream encipher is applied to the image to obtain the final marked encrypted image $\mathbb{M}$.

In order to achieve a higher level of security, we encrypt the secret data using $\mathcal{K}_d$ before embedding them into the encrypted image. Any existing data encryption algorithm can be used to encrypt the secret data. Thus, the content of secret data is protected. In addition, we shuffle the image blocks in the encrypted image using $\mathcal{K}_d$ before performing the data embedding process. This changes the data embedding orders in the image blocks.

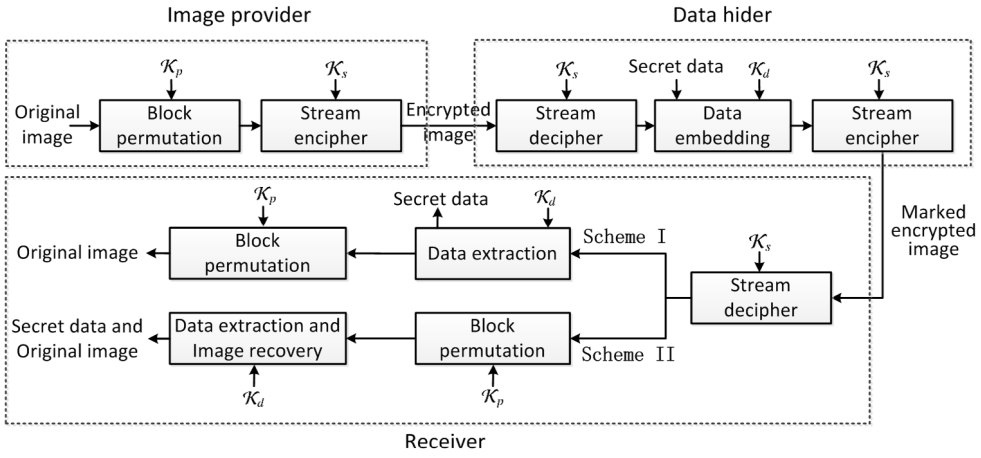


Fig. 4. Framework of the proposed ABPEE-RDHEI.

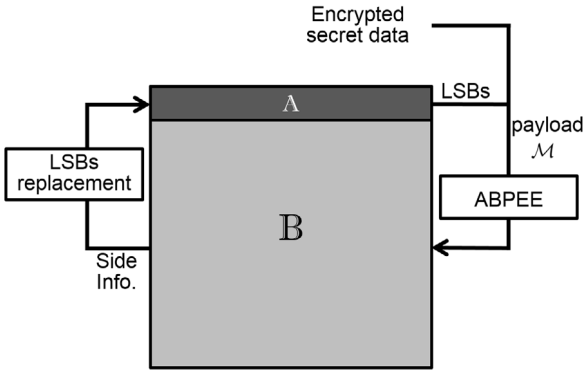


Fig. 5. Image partition and self-contained embedding.

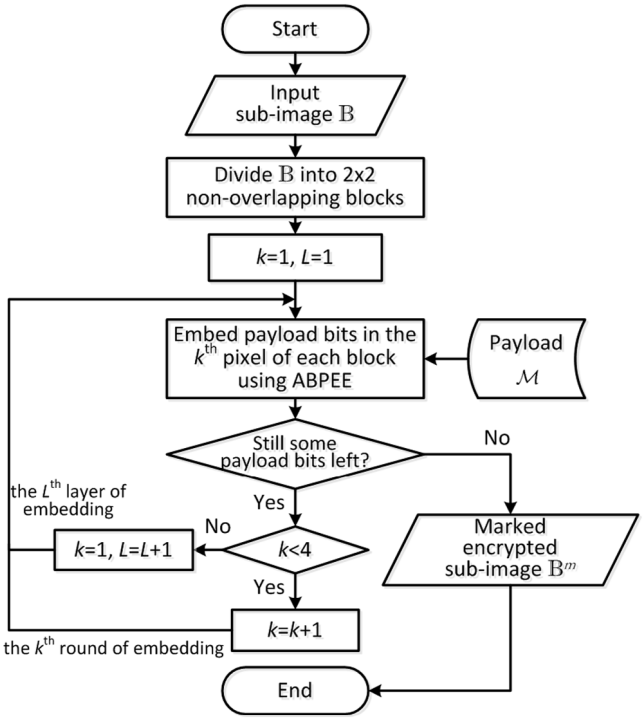


Fig. 6. Flowchart of payload embedding.

Because overflow may happen after secret data embedding, and some parameters will also be generated, we use a self-contained embedding strategy to store these information as shown in Fig. 5. The shuffled image is first separated into two sub-images namely $\mathbb{A}$ and $\mathbb{B}$, where $\mathbb{A}$ and $\mathbb{B}$ contain the first two rows and the remaining (N_1-2) rows of the shuffled image, respectively. The encrypted secret data and LSBs of sub-image $\mathbb{A}$ are concatenated to form payload $\mathcal{M}$ to be embedded into sub-image $\mathbb{B}$ using an adaptive BPEE (ABPEE). After embedding payload $\mathcal{M}$, the side information, containing parameters and overflow information generated from sub-image $\mathbb{B}$, will be stored into LSBs of sub-image $\mathbb{A}$ using bit replacement. Finally, all image blocks are unshuffled using key $\mathcal{K}_d$ to generate the marked encrypted image. Next, we will present payload embedding in detail.

The flowchart of payload embedding is shown in Fig. 6. The sub-image $\mathbb{B}$ is first divided into a series 2×2 non-overlapped image blocks. Then, payload bits are embedded into the image blocks using ABPEE. Four rounds of embedding form a layer of embedding. The iterative embedding processes will stop when all payload bits are completely embedded.

ABPEE is an adaptive strategy of BPEE. In each embedding round, ABPEE only selects the pixels in smooth regions in the image to embed payload bits while keeping the pixels in fine-grained regions unchanged. Thus, it can achieve a relatively high image quality in the directly recovered image. In addition, ABPEE can effectively reduce the overflow map size. More discussions can be found in Section 5.

4.2.1. Adaptive pixel selection

For an image block, we calculate two invariant features, C_b and $\hat{x}$ to determine whether the selected pixel x can be embedded with payload bits, where C_b denotes the complexity of the image block and is calculated by

$$C_b = \max\{x_r, x_c, x_d\} - \min\{x_r, x_c, x_d\} \tag{11}$$

and $\hat{x}$ is the prediction value of x that is calculated by BLP using Eq. (2).

Then, given two thresholds P and T, if $C_b < T$ and $\hat{x} \in [P, 255-P]$, pixel x in the current image block will be selected to embed payload bits; otherwise, keep it unchanged. The threshold P is calculated by Eq. (12) and the values of T are given in Table 1.

$$P = \begin{cases} 3 & \text{if } G \leqslant 4\times 10^{-4} \\ 5 & \text{if } 4\times 10^{-4} < G \leqslant 4\times 10^{-3} \\ 8 & \text{if } 4\times 10^{-3} < G \leqslant 1\times 10^{-2} \\ 10 & \text{otherwise} \end{cases} \tag{12}$$

Table 1
Threshold T under different values of embedding rate r and complexity $C_{\mathbb{B}}$ of sub-image $\mathbb{B}$.

T		$C_{\mathbb{B}}$							
		(0, 0.2]	(0.2, 0.3]	(0.3, 0.4]	(0.4, 0.5]	(0.5, 0.6]	(0.6, 0.7]	(0.7, 0.8]	(0.8, 1]
	⩽0.02	10	8	4	3	3	1	1	1
	(0.02, 0.05]	20	18	5	5	7	3	2	1
	(0.05, 0.15]	40	29	25	11	9	7	3	2
r (bpp)	(0.15, 0.25]	45	36	36	17	17	16	4	2
	(0.25, 0.35]	50	50	40	19	19	18	9	4
	(0.35, 0.45]	50	50	40	22	20	18	11	7
	>0.45	50	50	40	33	27	23	14	10

where

$$G = \frac{\sum_{i=1}^{N_1-2}\sum_{j=1}^{N_2} Q^{i,j}}{(N_1-2)N_2} \tag{13}$$

$$Q^{i,j} = \begin{cases} 0 & \text{if } \mathbb{B}^{i,j} \in [10,245] \\ 1 & \text{otherwise} \end{cases} \tag{14}$$

and (i, j) is the pixel location index.

Two factors will affect the value of T: the embedding rate r (.bpp) and the complexity $C_{\mathbb{B}}$ of sub-image $\mathbb{B}$, where the embedding rate r is given by the data hider. To calculate $C_{\mathbb{B}}$, we let b_j^k be the kth pixel in the jth block in sub-image $\mathbb{B}$, where $1 \leqslant k \leqslant 4$, $1 \leqslant j \leqslant S$ and $S = (N_1-2)N_2/4$. The average value of each block in $\mathbb{B}$ is then calculated by

$$v_j = \sum_{k=1}^{4} b_j^k/4 \tag{15}$$

Then we calculate the prediction-error values of $\mathbb{B}$ by

$$e_j^k = b_j^k - v_j, \quad \text{for } b_j^k \in [P, 255-P] \tag{16}$$

We concatenate all values of e_j^k to form a prediction-error sequence $\ddot{\mathbf{E}}$, and finally calculate the complexity $C_{\mathbb{B}}$ by

$$C_{\mathbb{B}} = \sum_{e=-3}^{2} h_{\ddot{\mathbf{E}}}(e)/(4S) \tag{17}$$

where $h_{\mathbf{A}}(t)$ is the same function that is used in Eq. (4).

4.2.2. ABPEE

In a single round of embedding, ABPEE divides the sequence with S pixels into two groups: Group-I and Group-II. Group-I contains the pixels that are selected by the previous proposed adaptive pixel selection method, and Group-II contains the remaining pixels. Then pixels in Group-I are utilized for payload embedding using BPEE, and pixels in Group-II are kept unchanged.

For the last round of embedding, the remaining Z bits of the payload may be less than $(h_{\ddot{\mathbf{E}}}(T_l^k)+h_{\ddot{\mathbf{E}}}(T_r^k))$, where $\ddot{\mathbf{E}}$ denotes the set of prediction-error values in Group-I. Thus, the embedding procedure will stop when all payload bits are successfully embedded, and the value of Z will be stored as a part of the side information for payload extraction at the receiver side.

4.2.3. Side information

After embedding the payload bits, the side information is then stored into the LSBs of sub-image $\mathbb{A}$ using bit replacement. The side information contains the following:

- the adaptive information;
- the number of rounds and layers of embedding;
- the prediction-error values that containing embedded bits;
- the size of payload bits embedded in the last embedding round;
- the information to solve the overflow problem.

In our proposed method, two thresholds, T and P, are utilized to indicate the adaptive information. The parameters k and L denote the number of rounds and layers for embedding, respectively. In each embedding round, T_l^ks and T_r^ks are the prediction-error values that contain embedded bits. Each embedding round and layer may obtain different values of T_l^k and T_r^k. Thus, we use $\{T_l^{k,j}\}_{j=1}^{L}$ and $\{T_r^{k,j}\}_{j=1}^{L}$ to denote the prediction-error values that contained embedded bits in the kth round and jth embedding layer. Finally, the value Z is utilized to denote the number of bits that embedded in the last round of embedding. These thresholds and parameters are stored to form side information.

After payload embedding, pixel values in sub-image $\mathbb{B}$ may exceed the range of [0,255]. We use a binary map to record information to solve the overflow problem. E.g., when the minimum pixel value of $\mathbb{B}$ becomes V, where $V < 0$. We convert the pixel values from $[V, -1]$ to $[0, -V-1]$ using one-to-one mapping (e.g., $V \to 0$, $(V+1) \to 1$, ...). Thus, when a pixel value falls into $[0, V-1]$, it is either an original pixel or a converted one. A binary map O_{map} is then generated to distinguish between original and converted (e.g., 0 for the original and 1 for the converted). The similar procedure can be applied to record the overflow information when pixel values larger than 255. The minimum (maximum) pixel value is also stored as a part of the side information for data extraction and image recovery at the receiver side.

4.3. Data extraction and image recovery

At the receiver side, we propose two schemes for data extraction and image recovery as shown in Fig. 4: Scheme I: data extraction before image recovering and Scheme II: data extraction after image recovering. Firstly, the stream decipher is applied to the marked encrypted image using $\mathcal{K}_s$. We denote the obtained image as $\hat{\mathbb{M}}$. Next, we introduce these two schemes of data extraction and image recovery detail.

4.3.1. Scheme I

(1) Data extraction

When holding security key $\mathcal{K}_d$, the receiver can extract the secret data from $\hat{\mathbb{M}}$ directly. Firstly, $\hat{\mathbb{M}}$ is divided into a number of 2×2 image blocks and these image blocks are scrambled by $\mathcal{K}_d$. The scrambled image is then separated into two sub-images, namely $\mathbb{A}'$ and $\mathbb{B}'$, by the same way in the data embedding process. Then, we extract parameters $Z, k, L, P, T, \{T_l^{k,j}\}_{j=1}^{L}, \{T_r^{k,j}\}_{j=1}^{L}$ and overflow map O_{map} from the LSBs of sub-image $\mathbb{A}'$. After modifying the pixel values in sub-image $\mathbb{B}'$ based on O_{map}, we can extract the payload from sub-image $\mathbb{B}'$ without any error. Note that data extraction follows the inverse order of embedding procedures from the last layer to the first layer, and from $\tilde{x}^4$ to $\tilde{x}^1$.

Here we take a single round of extraction as an example to describe the detailed payload extraction procedures. For each pixel $\tilde{x}$, we can obtain the same invariant features from its remaining three neighboring pixels to identify which group it belongs to using the obtained threshold values P and T. Firstly, a sequence with S pixels in a single extraction round is divided into two groups using the same pixel selection method in data embedding procedure. Then we extract payload bits from pixels in Group-I using Eqs. (7) and (8), and recover these pixels using Eqs. (9) and (10). Pixels in Group-II will be kept unchanged.

After extracting payload bits from the kth pixel of each block, go to the $(k-1)$th pixels for the next round of extraction, and the recovered

pixels will be considered as a part of the reference pixels to predict the $(k-1)$th pixels. Note that, in the first round of extraction, the data extraction and pixel recovering procedures will stop when Z bits of the payload are successfully obtained.

Then, the encrypted secret data are extracted from the payload and decrypted by $\mathcal{K}_d$. The original secret data are obtained. After obtaining the secret data from the payload, we recover the LSBs of sub-image $\mathbb{A}$ by replacing with the remaining payload bits. Then, all image blocks are shuffled by $\mathcal{K}_d$ and obtain the image $\hat{\mathbb{I}}$.

(2) Image recovery

After extracted the data, the receiver further unscrambles the whole image blocks using key $\mathcal{K}_p$ to obtain the final decrypted image. As each data extraction and image recovery step is reversible, the final decrypted image is exactly the same with the original one.

4.3.2. Scheme II

In some scenarios, the Cloud provider may want to embed some source information to the encrypted image. After the image being decrypted, the receiver also hope to trace the source of the image. Thus, Scheme II is introduced to perform data extraction after image decryption. The detailed procedures of Scheme II are described as follows.

The receiver first divides $\hat{\mathbb{M}}$ into 2×2 non-overlapped blocks. These blocks are unscrambled using $\mathcal{K}_p$ to generate the marked decrypted image. As the pixels in image blocks will have tiny changes after data embedding, the generated marked decrypted image is similar to the original one. The results will be experimentally demonstrated in Section 6.

Using both keys $\mathcal{K}_d$ and $\mathcal{K}_p$, one can further extract the secret data from the marked decrypted image. Because the permutation only changes the locations of image blocks and pixel within block, using $\mathcal{K}_p$ and $\mathcal{K}_d$, we first unshuffle image blocks to obtain the correct image block order for data extraction, and denote the first $N_2/2$ blocks as sub-image $\mathbb{A}'$ and the remaining image blocks as sub-image $\mathbb{B}'$. Then, we extract the side information from the LSBs of $\mathbb{A}'$ and reset the overflow pixel values in $\mathbb{B}'$ according to the overflow information, which is obtained from the side information. The payload extraction follows the same way as proposed in Scheme I. After obtaining the secret data and recovering the LSBs of sub-image $\mathbb{A}'$, all image blocks are unscrambled using $\mathcal{K}_p$ and $\mathcal{K}_d$. The original image is successfully recovered.

Due to the reversibility of ABPEE and the self-contained embedding protocol, the original image and secret data can be completely recovered without any error.

5. Discussions and analysis

In this section, we analyze the proposed ABPEE-RDHEI in terms of embedding rate, performance and security.

5.1. Embedding rate

Because the content owner does not reserve spare space before image encryption, the data hider has the flexibility to determine the size of secret data to be embedded. The maximum embedding rate of ABPEE-RDHEI depends on the smoothness of the original image. The more smooth the original image is, the larger maximum embedding rate it can achieve. In addition, the side information should be completely embedded into the LSBs of sub-image $\mathbb{A}$. For example, the side information has a length no more than $2N_2$ bits. One may enlarge the size of sub-image $\mathbb{A}$ or use more LSB planes to embed the side information. However, enlarging the size of $\mathbb{A}$ will reduce the size of sub-image $\mathbb{B}$ and thus result in a lower embedding rate. Using more LSB planes in $\mathbb{A}$ to embed side information may yield the marked decrypted image with low quality.

For simplicity, in this paper, the sub-image $\mathbb{A}$ contains only two rows of the image and only the LSBs in $\mathbb{A}$ are adopted to embed side information. However, when a larger number of pixels in the original image are close to 0 and/or 255, after embedding the payload bits, the size of the generated overflow map may exceed $2N_2$. For example, the image *Pepperes*[1] in Fig. 7(a) contains 5381 pixels whose values are less than 5 (within the red rectangle box in Fig. 7(b)). After being embedded with 0.3 bpp of secret data, an overflow map with 1486 bits is generated. Its size exceeds 1024 (the number of LSBs of sub-image $\mathbb{A}$). Due to the adaptive strategy in the proposed ABPEE-RDHEI, the maximum embedding rate of image *Pepperes* can reach 0.4 bpp. It is twice larger than those without the adaptive strategy.

5.2. Performance analysis

From Eqs. (5) and (6), the pixel values are either unchanged or modified by 1 when their prediction-error e_i^k equals to T_l^k (or T_r^k); when e_i^k is within the range of (T_l^k, T_r^k), the corresponding pixels will be kept unmodified; when e_i^k is without the range of $[T_l^k, T_r^k]$, the corresponding pixels will be shifted by 1. Thus, we can obtain the expected value of the mean square error (MSE) for a single embedding layer of ABPEE by

$$\frac{\sum \mathrm{E}(|\tilde{x}_i - x_i|^2)}{4S} \approx \frac{0.5\times N_c + N_s}{4S},\quad i = 1, 2, \ldots, 4S \tag{18}$$

where N_c and N_s are the capacity and amount of shifted pixels in a single embedding layer, respectively. Here the capacity N_c equals to the number of pixels when their prediction-error values belong to $\{T_l^k, T_r^k\}$. From Eq. (18), we can observe that, for a fixed capacity, the MSE will be small if N_s is small. In each embedding layer of our proposed ABPEE, only the pixels in smooth regions and have high probability to be the non-boundary pixels are selected to embed the payload bits. Thus, smaller MSE values will be obtained and this can result in the marked decrypted images with higher visual quality.

Here, we use the normalized prediction-error histograms derived from ABPEE and BPEE to demonstrate their performance, and the results are shown in Fig. 8. As an example, we only calculate the prediction-error histogram of the first embedding round in the first embedding layer. For ABPEE, the normalized prediction-error histogram is calculated by

$$\frac{h_{\check{\mathbf{E}}}(e)}{g},\quad \forall e \in \mathbb{Z} \tag{19}$$

and for BPEE it is calculated by

$$\frac{4h_{\dot{\mathbf{E}}}(e)}{S},\quad \forall e \in \mathbb{Z} \tag{20}$$

where $\check{\mathbf{E}}$ and $\dot{\mathbf{E}}$ are the sets of prediction-error values derived from ABPEE and BPEE, respectively, and g is length of $\check{\mathbf{E}}$. One can see from the results that ABPEE generates a sharper normalized prediction-error histogram than BPEE without the pixel selection procedure. This means that, to embed a fixed length of secret data, ABPEE shifts less pixels than BPEE. Thus, ABPEE can generate the marked decrypted image with better quality.

5.3. Security analysis

In RDHEI, the content owner/data hider does not allow the unauthorized user to access the original image/secret data. Thus both the original image and secret data need to be protected. In our proposed ABPEE-RDHEI, we use the block-level permutation and stream cipher to encrypt the original image. For an $N_1\times N_2$ image divided into N blocks, there will be totally $N!4^N$ possible permutations. For the stream cipher, there are $256^{N_1N_2}$ possible bit sequences to change the pixel values of the image. Thus, the possibility of breaking the encrypted results without $\mathcal{K}_p$ and $\mathcal{K}_s$ is as small as $\frac{1}{256^{N_1N_2}N!4^N}$. Although the permutation process may reveal the original image histogram information to the data hider, the possibility of obtaining the original image is as small as $\frac{1}{N!4^N}$. To protect the secret data, using $\mathcal{K}_d$, we first encrypt it and shuffle the data embedding order. Then, the stream cipher is applied to change the bits in the whole image. If the possibility of successfully recovering the secret data without $\mathcal{K}_d$ is $\frac{1}{W}$, the possibility of breaking the embedded secret data bits without the embedding key $\mathcal{K}_d$ is as low as $\frac{1}{256^{N_1N_2}WN!4^N}$.

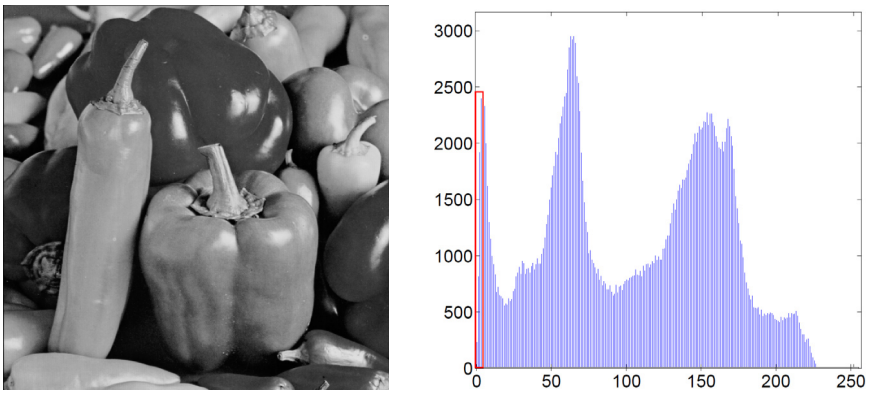


Fig. 7. An example of the original image. (a) *Pepperes* with size of 512 × 512 and its (b) histogram.

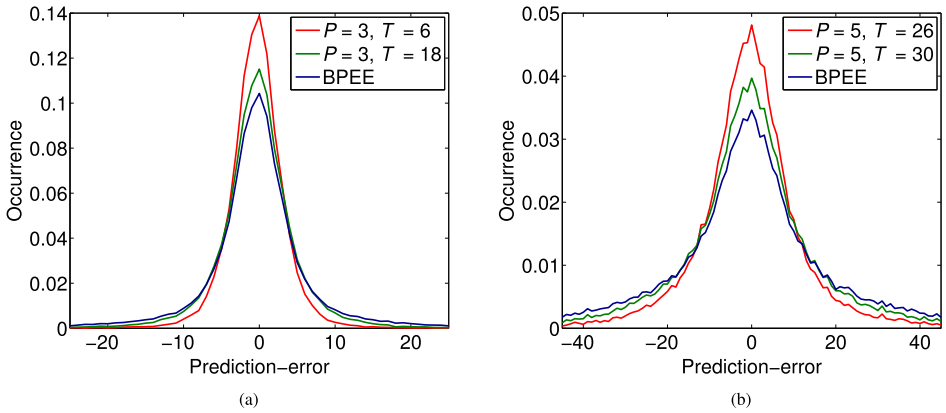


Fig. 8. Normalized prediction-error histograms derived after the first embedding round and first embedding layer of images (a) *Lena* and (b) *Baboon* using ABPEE with pixel selection (red and green lines) and BPEE without pixel selection (blue line), respectively.

6. Simulation results and comparisons

We take the *Lena* image to demonstrate the proposed ABPEE-RDHEI algorithm, and the results are shown in Fig. 9. We can observe that, even being embedded with 0.3 bpp secret data, the marked decrypted image (Fig. 9(c)) has high visual quality with a PSNR value of 44.89 dB. And we can also obtain a losslessly recovered image (Fig. 9(d)).

6.1. Comparison of embedding strategies

To test the embedding performance of ABPEE, we select 6 test images from the database[1] to compare the proposed algorithm under three different embedding strategies: ABPEE, BPEE, and MED based PEE. They are denoted as ABPEE-RDHEI, BPEE-RDHEI and MED-RDHEI. BPEE-RDHEI follows the same procedure with ABPEE-RDHEI without pixel selection. For MED-RDHEI, we use MED to replace BLP for the target pixel prediction, and other procedures follow the same way as ABPEE-RDHEI. Table 2 compares the PSNR values of their marked decrypted images under various embedding rates. We can observe that ABPEE-RDHEI outperforms other two methods under all embedding rates. Table 3 lists the lengths of overflow maps generated by these three algorithms under different embedding rates. Compared with the other two methods, ABPEE-RDHEI can significantly reduce the size of the overflow map especially when the image contains a large number of pixels approach to 0 and/or 255 (e.g., *Peppers*).

In order to quantitatively measure the performance of ABPEE-RDHEI, we randomly select 6 groups of images from BOWSBase.[2] Each group contains 200 images. Then, we apply ABPEE-RDHEI, BPEE-RDHEI and MED-RDHEI on each group of images with different embedding rates. Fig. 10 plots the PSNR results of their marked decrypted images. The results show that ABPEE-RDHEI outperforms other two methods in most cases. Fig. 11 plots the average PSNR values of 200 marked decrypted images with different embedding rates. One can observe that the difference of the average PSNRs between ABPEE-RDHEI and BPEE-RDHEI decreases with the increasing of the embedding rate. This is because when more secret data are embedded, a larger value of T is utilized. Thus, in each embedding round, more pixels are selected to Group-I while less pixels will be in Group-II. When the number of pixels in Group-II approaches to 0, the performance of ABPEE-RDHEI is close to that of BPEE-RDHEI. Compared with ABPEE-RDHEI and MED-RDHEI, the difference of the average PSNR increases with the decreasing of embedding rates. This shows that BLP can obtain more precise prediction results than MED, especially when more embedding layers are used.

6.2. Performance comparison

We select 3 commonly used test images (*Lena, Baboon, Airplane*) from the database[1] to compare the performance of ABPEE-RDHEI with several existing VRAE methods. Fig. 12 plots the PSNR results of different marked decrypted images generated by these algorithms under given embedding rates. For methods in [12,20,21], we set the block size as 8 × 8 for demonstration. Because the data extraction of these methods is based on the smoothness of the original image, incorrect data bits may be extracted when the original image contains many textures. Thus, the pure capacity C (bits) reduces to $C(1 - H(\rho))$, where $H(\rho)$ is the

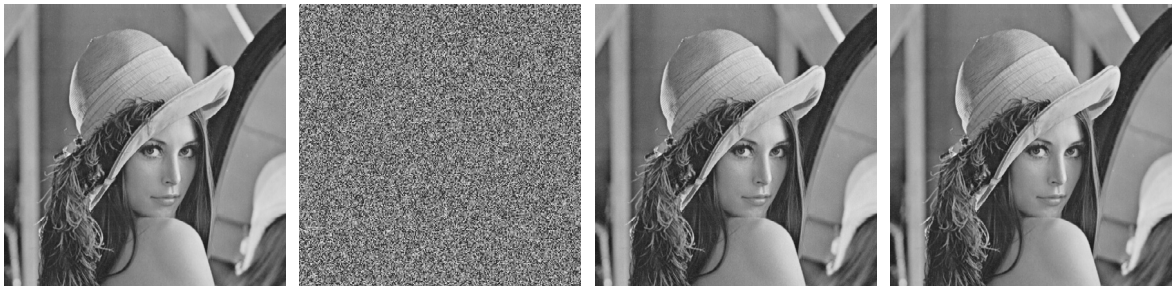
[2] http://bows2.ec-lille.fr/.

Table 2
PSNR comparisons of marked decrypted images generated by BPEE-RDHEI, MED-RDHEI and ABPEE-RDHEI under various embedding rates.

PSNR results (dB)												
Embedding rate (bpp)		0.005	0.02	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.5
Lena	BPEE-RDHEI	62.18	57.87	54.32	51.49	49.75	48.13	45.81	44.32	43.12	41.74	39.65
	MED-RDHEI	64.23	58.52	55.50	52.19	49.70	46.81	45.00	43.28	41.82	40.27	37.30
	ABPEE-RDHEI	64.89	58.78	55.93	52.71	50.31	48.45	46.27	44.89	43.63	42.30	39.86
Airplane	BPEE-RDHEI	64.34	60.11	56.58	53.67	51.90	50.59	49.60	48.60	46.70	45.32	43.03
	MED-RDHEI	67.25	63.00	58.55	54.54	52.79	50.03	48.43	47.14	45.32	43.50	41.13
	ABPEE-RDHEI	67.81	63.58	59.27	55.87	53.78	52.43	51.15	49.29	47.61	46.38	43.72
Barbara	BPEE-RDHEI	60.76	56.39	52.84	49.99	47.71	44.84	43.07	41.19	39.79	38.33	35.93
	MED-RDHEI	63.64	59.28	55.34	51.09	47.42	45.02	42.89	40.69	38.98	37.07	33.88
	ABPEE-RDHEI	63.93	59.32	55.60	52.10	48.87	46.32	44.43	42.72	41.10	39.75	36.46
Boat	BPEE-RDHEI	62.39	58.06	54.57	51.58	49.84	48.32	45.85	44.27	42.98	41.53	39.27
	MED-RDHEI	64.40	60.14	55.67	52.56	50.13	47.04	45.40	43.38	41.93	40.24	37.21
	ABPEE-RDHEI	65.02	60.70	56.46	53.26	50.82	49.04	46.79	45.21	43.80	42.50	39.56
Baboon	BPEE-RDHEI	57.19	52.99	49.38	44.10	40.86	38.35	35.90	34.14	32.50	31.01	–
	MED-RDHEI	60.03	54.85	49.77	44.11	40.18	37.03	34.30	32.32	30.42	–	–
	ABPEE-RDHEI	59.87	55.02	50.55	45.23	41.62	38.75	36.10	34.31	32.64	31.00	–
Peppers	BPEE-RDHEI	61.56	57.29	53.7	50.82	49.04	–	–	–	–	–	–
	MED-RDHEI	62.32	57.97	53.70	49.54	46.80	44.30	42.17	40.19	–	–	–
	ABPEE-RDHEI	62.28	58.57	54.10	51.06	48.57	46.26	44.64	42.61	41.26	40.10	–

Table 3
Lengths of the overflow maps of different images with various embedding rates.

Overflow map size (bits)												
Embedding rate (bpp)		0.005	0.02	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.5
Lena	BPEE-RDHEI	0	0	0	0	0	0	0	0	0	0	0
	MED-RDHEI	0	0	0	0	0	0	0	0	0	0	0
	ABPEE-RDHEI	0	0	0	0	0	0	0	0	0	0	0
Airplane	BPEE-RDHEI	0	0	0	0	0	0	0	0	0	0	0
	MED-RDHEI	0	0	0	0	0	0	0	0	0	0	0
	ABPEE-RDHEI	0	0	0	0	0	0	0	0	0	0	0
Barbara	BPEE-RDHEI	0	0	0	0	0	0	0	0	0	0	0
	MED-RDHEI	0	0	0	0	0	0	0	0	0	0	0
	ABPEE-RDHEI	0	0	0	0	0	0	0	0	0	0	0
Boat	BPEE-RDHEI	0	0	0	0	0	27	27	27	27	27	27
	MED-RDHEI	0	0	0	0	0	0	0	0	0	0	0
	ABPEE-RDHEI	0	0	0	0	0	0	0	0	0	0	0
Baboon	BPEE-RDHEI	0	0	76	121	157	233	275	311	368	522	956
	MED-RDHEI	0	0	0	0	0	87	178	186	356	550	2098
	ABPEE-RDHEI	0	0	0	0	0	0	89	126	190	317	844
Peppers	BPEE-RDHEI	61	76	106	160	233	1027	1255	1486	3058	3291	5641
	MED-RDHEI	0	0	0	0	0	59	66	394	1443	3687	9234
	ABPEE-RDHEI	0	0	0	0	0	0	0	74	93	463	1735



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

Fig. 9. Example of the proposed ABPEE-RDHEI algorithm. (a) The original image; (b) the marked encrypted image with a embedding rate of $r = 0.3$ bpp; (c) the marked decrypted image, PSNR = 44.89 dB; (d) the recovered image.

binary entropy function with the error rate ρ. For other methods, we choose the results only when the original image can be successfully recovered without any error. In [26] and [30], the block size is set as 4×4. For [30], the 6 LSBs of each pixel in blocks are utilized to embed secret data. For methods in [40,41], we set the block size to 2×2 for demonstration. From those results we can observe that our proposed ABPEE-RDHEI obtains the best visual quality of the marked decrypted image compared, with other methods in most cases. Compared with the Yin et al.'s method [26], although ABPEE-RDHEI has slightly higher PSNR values when the embedding rates are relatively small, ABPEE-RDHEI is able to embed a larger size of secret data than Yin et al.'s method. Method in [40] is also based on block operation. It uses the

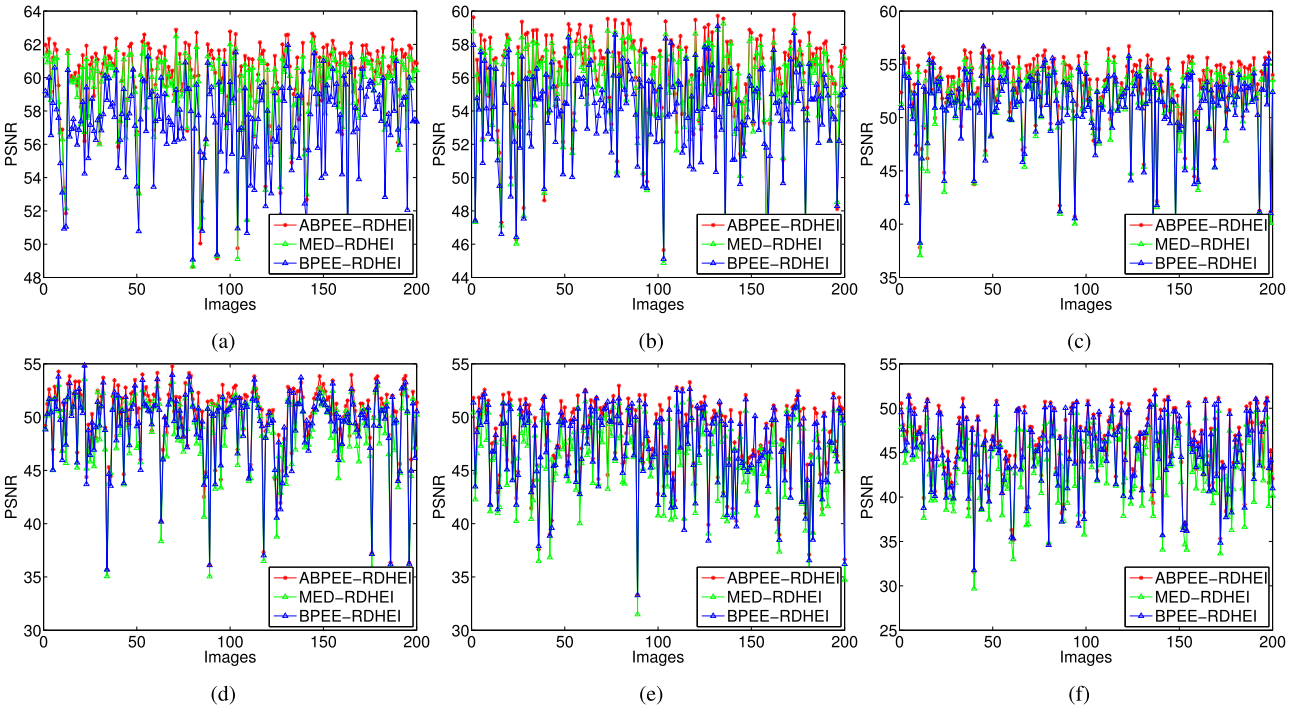


Fig. 10. PSNR results of 200 marked decrypted images generated by ABPEE-RDHEI, MED-RDHEI and BPEE-RDHEI under different embedding rates. (a) $r = 0.05$; (b) $r = 0.1$; (c) $r = 0.2$; (d) $r = 0.3$; (e) $r = 0.4$; (f) $r = 0.5$.

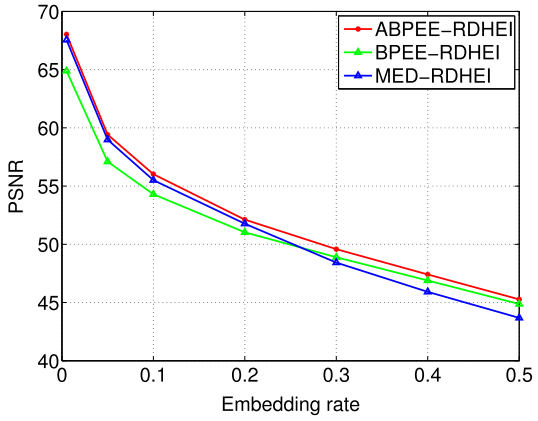


Fig. 11. Average PSNR values of 200 marked decrypted images generated by ABPEE-RDHEI, MED-RDHEI and BPEE-RDHEI under different embedding rates.

permutation and stream cipher to encrypt image blocks, while the proposed scheme uses the block permutation for block scrambling and stream cipher for encrypting the whole image. In addition, their data hiding processes are also different. Method in [40] treats the encrypted image as the original cover image, and uses the traditional RDH methods (e.g., DHE and PEHS) to embed data. Due to the significant changes of pixel values between image blocks, the imprecise pixel prediction values result in relatively low embedding rates. In our proposed scheme, the prediction process considers only pixels within the block. Therefore, more precise prediction results will be obtained. Moreover, due to the iterative embedding, the embedding rate is significantly improved.

7. Conclusion

In this paper, we proposed a new predictor called BLP to predict pixels within a 2×2 image block. Using BLP, we proposed a reversible data hiding method called BPEE to embed secret data into an image block by block. Then, we further proposed an adaptive BPEE for reversible data hiding in encrypted images called ABPEE-RDHEI. It inherits all merits of the S-VRAE methods and achieves full reversibility. In addition,

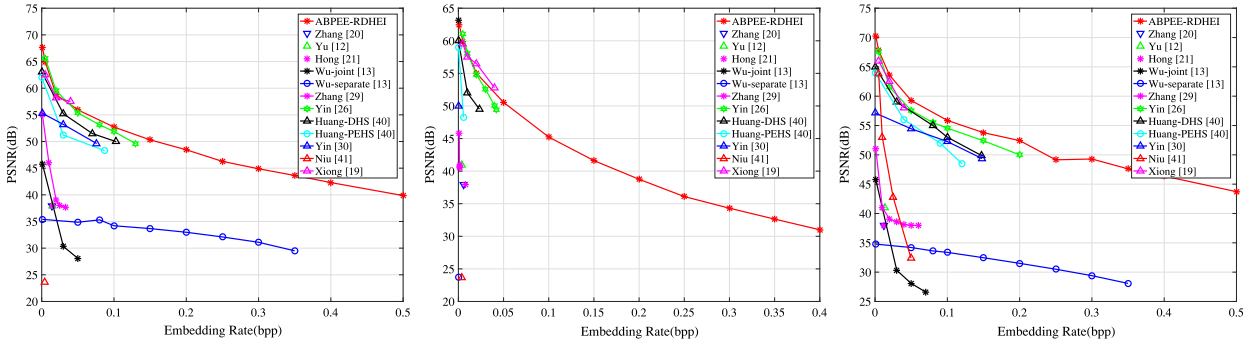


Fig. 12. PSNR results of marked decrypted images generated by ABPEE-RDHEI and other existing methods under different embedding rates. (a) *Lena*; (b) *Baboon* and (c) *Airplane*.

it is able to extract the secret data before or after image decryption to adapt different applications. Experimental and comparison results shown that the proposed ABPEE-RDHEI has excellent performance and can out perform several state-of-the-art VRAE methods in terms of the embedding rate and quality of the marked decrypted images.

Acknowledgments

This work was supported in part by the Macau Science and Technology Development Fund under Grant FDCT/016/2015/A1 and by the Research Committee at University of Macau under Grant MYRG2016-00123-FST.

References

[1] J. Tian, Reversible data embedding using a difference expansion, IEEE Trans. Circuits Syst. Video Technol. 13 (8) (2003) 890–896.
[2] Z. Ni, Y.-Q. Shi, N. Ansari, W. Su, Reversible data hiding, IEEE Trans. Circuits Syst. Video Technol. 16 (3) (2006) 354–362.
[3] X. Li, B. Li, B. Yang, T. Zeng, General framework to histogram-shifting-based reversible data hiding, IEEE Trans. Image Process. 22 (6) (2013) 2181–2191.
[4] X. Li, W. Zhang, X. Gui, B. Yang, Efficient reversible data hiding based on multiple histograms modification, IEEE Trans. Inf. Forensics Secur. 10 (9) (2015) 2016–2027.
[5] X. Zhang, S. Wang, Efficient steganographic embedding by exploiting modification direction, IEEE Commun. Lett. 10 (11) (2006) 781–783.
[6] C. Qin, C.-C. Chang, T.-J. Hsu, Reversible data hiding scheme based on exploiting modification direction with two steganographic images, Multimedia Tools Appl. 74 (15) (2015) 5861–5872.
[7] D.M. Thodi, J.J. Rodriguez, Expansion embedding techniques for reversible watermarking, IEEE Trans. Image Process. 16 (3) (2007) 721–730.
[8] X. Li, B. Yang, T. Zeng, Efficient reversible watermarking based on adaptive prediction-error expansion and pixel selection, IEEE Trans. Image Process. 20 (12) (2011) 3524–3533.
[9] Z. Qian, X. Zhang, S. Wang, Reversible data hiding in encrypted JPEG bitstream, IEEE Trans. Multimedia 16 (5) (2014) 1486–1491.
[10] K. Ma, W. Zhang, X. Zhao, N. Yu, F. Li, Reversible data hiding in encrypted images by reserving room before encryption, IEEE Trans. Inf. Forensics Secur. 8 (3) (2013) 553–562.
[11] W. Puech, M. Chaumont, O. Strauss, A reversible data hiding method for encrypted images, security, forensics, steganography, and watermarking of multimedia contents X, Proc. SPIE 6819 (2008).
[12] J. Yu, G. Zhu, X. Li, J. Yang, An improved algorithm for reversible data hiding in encrypted image, in: Digital Forensics and Watermaking, in: Lecture Notes in Computer Science, vol. 7809, 2013, pp. 384–394.
[13] X. Wu, W. Sun, High-capacity reversible data hiding in encrypted images by prediction error, Signal Process. 104 (2014) 387–400.
[14] M. Li, D. Xiao, Z. Peng, H. Nan, A modified reversible data hiding in encrypted images using random diffusion and accurate prediction, ETRI J. 36 (2) (2014) 325–328.
[15] C. Qin, X. Zhang, Effective reversible data hiding in encrypted image with privacy protection for image content, J. Vis. Commun. Image Represent. 31 (2015) 154–164.
[16] J. Zhou, W. Sun, L. Dong, X. Liu, O.C. Au, Y.Y. Tang, Secure reversible image data hiding over encrypted domain via key modulation, IEEE Trans. Circuits Syst. Video Technol. 26 (3) (2016) 441–452.
[17] R. Jose, G. Abraham, A separable reversible data hiding in encrypted image with improved performance, in: International Conference on Microelectronics, Communications and Renewable Energy AICERA/ICMiCR, 2013, pp. 1–5.
[18] X. Zhang, C. Qin, Guangling, Reversible data hiding in encrypted images using pseudorandom sequence modulation, Digital Forensics Watermaking 7809 (2013) 358–367.
[19] L. Xiong, Z. Xu, Y.-Q. Shi, An integer wavelet transform based scheme for reversible data hiding in encrypted images, Multidimens. Syst. Signal Process. (2017). http://dx.doi.org/10.1007/S11045-017-0497-5.
[20] X. Zhang, Reversible data hiding in encrypted image, IEEE Signal Process. Lett. 18 (4) (2011) 255–258.
[21] W. Hong, T.-S. Chen, H.-Y. Wu, An improved reversible data hiding in encrypted images using side match, IEEE Signal Process. Lett. 19 (4) (2012) 199–202.
[22] X. Liao, C. Shu, Reversible data hiding in encrypted images based on absolute mean difference of multiple neighboring pixels, J. Vis. Commun. Image Represent. 28 (2015) 21–27.
[23] Z. Qian, X. Han, X. Zhang, Separable reversible data hiding in encrypted images by n-nary histogram modification, in: The Third International Conference on Multimedia Technology, 2013, p. 8.
[24] S. Zhang, T. Gao, G. Sheng, A joint encryption and reversible data hiding scheme based on integer-dwt and arnold map permutation, J. Appl. Math. 2014 (2014) 12.
[25] S. Yi, Y. Zhou, Improved reversible data hiding in encrypted images using histogram modification, in: 2016 IEEE International Conference on Systems, Man and Cybernetics, 2016, pp. 1833–1837.
[26] Z. Yin, B. Luo, W. Hong, Separable and error-free reversible data hiding in encrypted image with high payload, Sci. World J. 2014 (2014) 8.
[27] X. Zhang, J. Long, Z. Wang, H. Cheng, Lossless and reversible data hiding in encrypted images with public-key cryptography, IEEE Trans. Circuits Syst. Video Technol. 26 (9) (2016) 1622–1631.
[28] X. Zhang, Z. Qian, G. Feng, Y. Ren, Efficient reversible data hiding in encrypted images, J. Vis. Commun. Image Represent. 25 (2) (2014) 322–328.
[29] X. Zhang, Separable reversible data hiding in encrypted image, IEEE Trans. Inf. Forensics Secur. 7 (2) (2012) 826–832.
[30] Z. Yin, A. Abel, J. Tang, X. Zhang, B. Luo, Reversible data hiding in encrypted images based on multi-level encryption and block histogram modification, Multimedia Tools Appl. 76 (3) (2017) 3899–3920.
[31] T. Mathew, M. Wilscy, Reversible data hiding in encrypted images by active block exchange and room reservation, in: 2014 International Conference on Contemporary Computing and Informatics, IC3I, 2014, pp. 839–844.
[32] W. Zhang, K. Ma, N. Yu, Reversibility improved data hiding in encrypted images, Signal Process. 94 (2014) 118–127.
[33] S. Yi, Y. Zhou, An improved reversible data hiding in encrypted images, in: 2015 IEEE China Summit and International Conference on Signal and Information Processing, ChinaSIP, 2015, pp. 225–229.
[34] 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. Cybernet. 46 (5) (2016) 1132–1143.
[35] M.J. Weinberger, G. Seroussi, G. Sapiro, The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS, IEEE Trans. Image Process. 9 (8) (2000) 1309–1324.
[36] X. Wu, N. Memon, Context-based, adaptive, lossless image coding, IEEE Trans. Commun. 45 (4) (1997) 437–444.
[37] M. Chen, Z. Chen, X. Zeng, X. Zhang, Reversible data hiding using additive prediction-error expansion, in: Proc. 11th ACM Workshop Multimedia and Seciruty, 2009, pp. 19–24.
[38] C. Qin, C.C. Chang, Y.H. Huang, L.T. Liao, An inpainting-assisted reversible steganographic scheme using a histogram shifting mechanism, IEEE Trans. Circuits Syst. Video Technol. 23 (7) (2013) 1109–1118.
[39] R.M. Rad, W. KokSheik, G. Jing-Ming, A unified data embedding and scrambling method, IEEE Trans. Image Process. 23 (4) (2014) 1463–1475.
[40] F. Huang, J. Huang, Y.Q. Shi, New framework for reversible data hiding in encrypted domain, IEEE Trans. Inf. Forensics Secur. 11 (12) (2016) 2777–2789.
[41] X. Niu, Z. Yin, X. Zhang, J. Tang, B. Luo, Reversible data hiding in encrypted AMBTC compressed images, in: Digital Forensics and Watermarking: 15th International Workshop, 2017, pp. 436–445.