Separable and Reversible Data Hiding in Encrypted Images Using Parametric Binary Tree Labeling

Shuang Yi, *Student Member, IEEE*, and Yicong Zhou, *Senior Member, IEEE*

***Abstract*—This paper first introduces a parametric binary tree labeling scheme (PBTL) to label image pixels in two different categories. Using PBTL, a data embedding method (PBTL–DE) is proposed to embed secret data to an image by exploiting spatial redundancy within small image blocks. We then apply PBTL–DE into the encrypted domain and propose a PBTL-based reversible data hiding method in encrypted images (PBTL–RDHEI). PBTL–RDHEI is a separable and reversible method that both the original image and secret data can be recovered and extracted losslessly and independently. Experiment results and analysis show that PBTL–RDHEI is able to achieve an average embedding rate as large as 1.752 bpp and 2.003 bpp when block size is set to 2×2 and 3×3, respectively.**

***Index Terms*—Reversible data hiding, encrypted images, parametric binary tree labeling scheme, privacy protection.**

I. Introduction

IMAGE encryption is a technique to transform the original meaningful image into a noise-like one to prevent unauthorized access [1]. Using the correct security key, users are able to decrypt the image to see its original content [2]. Modern cryptography mainly focus on the block cipher and stream cipher. Different from the image encryption, reversible data hiding (RDH) is a technique to embed secret data into cover images by slightly modifying their pixel values [3]. It aims to imperceptibly modify the cover image pixel values. Existing RDH methods are almost derived from the following three fundamental strategies: lossless compression [4], [5], histogram shifting (HS) [6] and difference expansion [7], [8]. Due to the reversible property, the original image can be fully recovered after extracting the secret data. This can be used in many applications such as medial and military image processing.

Recently, due to the development of cloud computing and cloud storage, many researchers show their interests in developing reversible data hiding method in encrypted images (RDHEI) [9]–[15]. In this scheme, there are three end users: the content-owner, data-hider and receiver. Here the content-owner is also regarded as the original image provider who encrypts the image before sending it to the data-hider. The data-hider embeds some secret data into the encrypted image and has no privilege to access the original image content. At the receiver end, we desire that users only with specific authority are able to access the original image, secret data or both. This scheme can be used in many privacy-preserving applications such as cloud storage, medical image management system [16], secure remote sensing system [9], etc. Take the cloud storage for example, in order to protect the original image content from been revealed to the public, the content-owner encrypts it before sending it to the cloud. Without knowing the original image content, the cloud administrator who acts as the data-hide wants to embed some additional data, e.g., the image source information, into the encrypted image for the purpose of image management. At the receiver side, according to different privileges, one can obtain the secret data or original image separately.

Existing RDHEI methods can be classified into three categories, namely reserving room before encryption (RRBE) [11], [17]–[20], reserving room after encryption (RRAE) [9], [16], [21] and vacating room by encryption (VRBE) [22]–[24]. The RRBE methods use some traditional RDH methods or exploit spatial redundancy from the original image to reserve spare room before image encryption. The second category mainly uses the standard image encryption algorithms such as AES, RC4 or homomorphic encryption to encrypt the original image directly. The VRBE methods adopt some specific encryption algorithm to encrypt the original image while keeping spatial redundancy in the encrypted image so that it can be exploited for data embedding.

Ma *et al.* [11] first proposed a RRBE method. They divide the original image into smooth area and coarse area, and embed several least significant bit (LSB) planes of the coarse area into the smooth area using the traditional RDH method [6]. The reserved LSB planes are then used for data embedding. Mathew *et al.* [20] improved Ma's method by divide the original image into small blocks and then separate these blocks into smooth and coarse areas for spare space reservation. Cao *et al.* [18] adopt the sparse representation technique to compress the small smooth image blocks to reserve room. Method in [17] randomly select some pixels from the original image and predict them by their surrounding pixels, the obtained prediction-error values are encrypted and reserved for data embedding. Yi *et al.* [19] improved Zhang's method [17] by predicting half of the pixels in the original image so that the embedding rate can be increased.

Manuscript received November 15, 2017; revised March 29, 2018; accepted May 12, 2018. Date of publication June 7, 2018; date of current version December 20, 2018. This work was supported in part by the Macau Science and Technology Development Fund under Grant FDCT/189/2017/A3, and in part by the Research Committee at University of Macau under Grant MYRG2016-00123-FST and Grant MYRG2018-00136-FST. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Balakrishnan Prabhakaran. *(Corresponding author: Yicong Zhou.)*

The authors are with the Department of Computer and Information Science, University of Macau, Macau 999078, China (e-mail: yishuang0227@gmail.com; yicongzhou@umac.mo).

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/TMM.2018.2844679

Although using the traditional RDH method to reserve room from the original image is more efficient and will obtain a larger embedding rate than RRAE methods, it may be impracticable. Because the content-owner have no idea about the size of the forthcoming data, or the content-owner may has no computational capacity to apply the RDH to reserve room [25]. Therefore, a lot of researchers are mainly focus on developing RRAE methods. Puech *et al.* [26] embed secret data into the AES-encrypted image and use local standard deviation to reconstruct the original image. Zhang [16] and Hong *et al.* [27] use the stream cipher to encrypt the original image and divide the encrypted image into equal sized small blocks. Then each block is embedded with one bit of the secret data by flipping the 3 LSBs of half of the pixels that randomly selected from the block. Li [28] modified Zhang's method [16] by flipping the pixels in the specific positions to embed data. Zhou *et al.* [9] use a public key modulation mechanism to embed secret data. In this method, one encrypted image block can embed more than one bit of the secret data. Thus, the embedding rate is significantly improved. At the receiver side, the SVM technique is adopt to extract the secret data and recover the original image. These are joint VRAE methods [9], [16], [27], [28] that the data extraction and image recovery are performed simultaneously. In addition, the data extraction and image recovery are based on analyzing the local smoothness of the directly decrypted image. Thus, they may suffer from incorrect extracted data and recovered image when block size is small. More practically, methods in Zhang *et al.* [21], [29] and Zheng *et al.* [30] suggest to develop separable VRAE methods by compressing some LSB planes in the stream cipher encrypted image to reserve room for data embedding. In this way, data extraction and image recovery can be performed separately. In [31], the pseudorandom sequence modulation mechanism is adopted to embed secret data. Wu *et al.* [32] embed secret data by replacing the specific bit positions in the two most significant bit (MSB) planes. Methods in [33]–[35] suggest to use the homomorphic encryption to encrypt the original image. However, these methods suffer from pixel expansion that the enlarged bit depth of pixels will result in a larger storage space.

Science reserving room from the stream cipher encrypted image is quite difficult and results in low embedding rate, some researchers try to develop VRBE method. Method in [22] divides the original image into blocks, then a coarse-grained permutation is used to change block locations within the image and a fine-grained permutation is applied to change pixel locations within the block. Data embedding is performed using the HS method in each block. This method suffers from information leakage in the encrypted image, because only the pixel locations are changed. Method in [23] uses a modulation operation to enhance the image encryption security. However, it is also limited in embedding rate and requires the pixel values of the encrypted image within the rage of [1, 254]. In Huang *et al.*'s method [24], the original image is encrypted by block permutation and block based bit-XOR. Then the traditional RDH method such as difference histogram shifting (DHS) and prediction-error histogram shifting (PHS) are utilized to embed secret data. In [36], stream cipher is applied to several MSB planes of the image, and HS is utilized to embed secret data into the rest LSB planes of each encrypted image block.

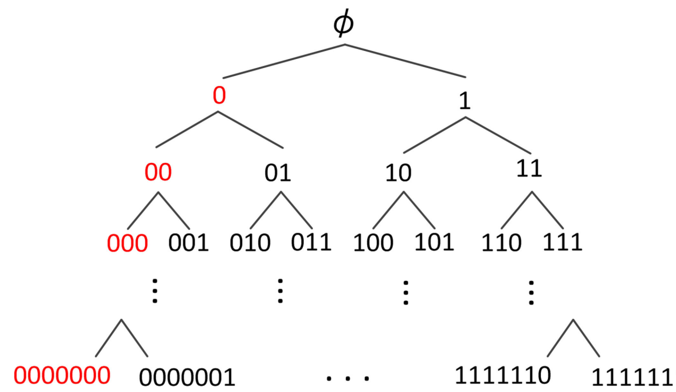


Fig. 1. Distribution of binary codes based on a full binary tree.

In this paper, we propose an RDHEI method using parametric binary tree labeling scheme (PBTL-RDHEI). It is a VRBE method that keeps spatial correlations within small encrypted image blocks, so that secret data embedding can be accomplished by exploiting the spatial redundancy from the encrypted image. Different from the methods in [22]–[24] that use the traditional RDH method to embed secret data, we adopt the PBTL based reversible data embedding so that the embedding rate can be significantly improved. The contributions of this paper are summarized as follows:

1) We propose a parametric binary tree labeling scheme (PBTL) to label pixels in two different categories. Selecting different settings of parameters, PBTL will provide different pixel labeling strategies.
2) Using PBTL, we propose a data embedding algorithm (PBTL-DE). It exploits spatial redundancy in small image blocks and embeds secret data into cover images using pixel labeling and bit replacement. Different from the traditional data embedding methods that embed secret data by modifying the plaintext cover image pixel values in an imperceptive way, PBTL-DE is designed for encrypted images. Thus, the significant changes to pixel values are acceptable.
3) Based on PBTL-DE algorithm, we further propose a PBTL-based RDHEI method (PBTL-RDHEI). Simulation results of applying PBTL-RDHEI to 1000 randomly selected test images demonstrate that PBTL-RDHEI is able to achieve an average embedding rate as large as 1.752 bpp and 2.003 bpp when block size is set to 2×2 and 3×3, respectively.

The rest of this paper is organized as follows: Section II proposes a parametric binary tree labeling scheme (PBTL). Using PBTL, a data embedding method is introduced in Section III. Section IV proposes the PBTL-RDHEI algorithm. Sections V analyze the redundancy preserving property of the image encryption method in PBTL-RDHEI. Section VI analyze the performance and security of PBTL-RDHEI. Section VII shows the experiment results and comparisons to some related works. Finally, Section VIII concludes this paper.

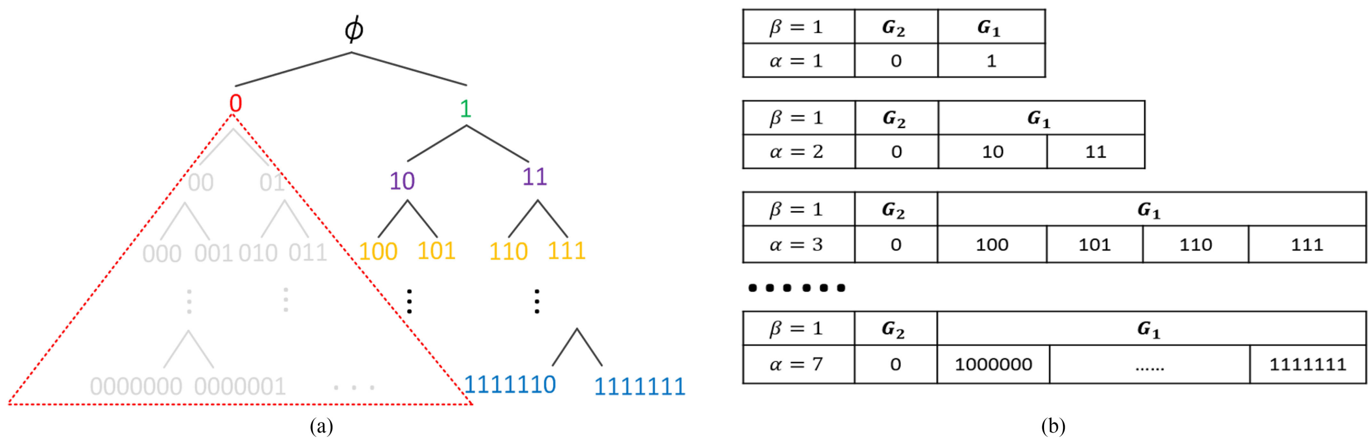


Fig. 2. Example of labeling bits selection when $\beta = 1$ and $\alpha = 1$ to 7.

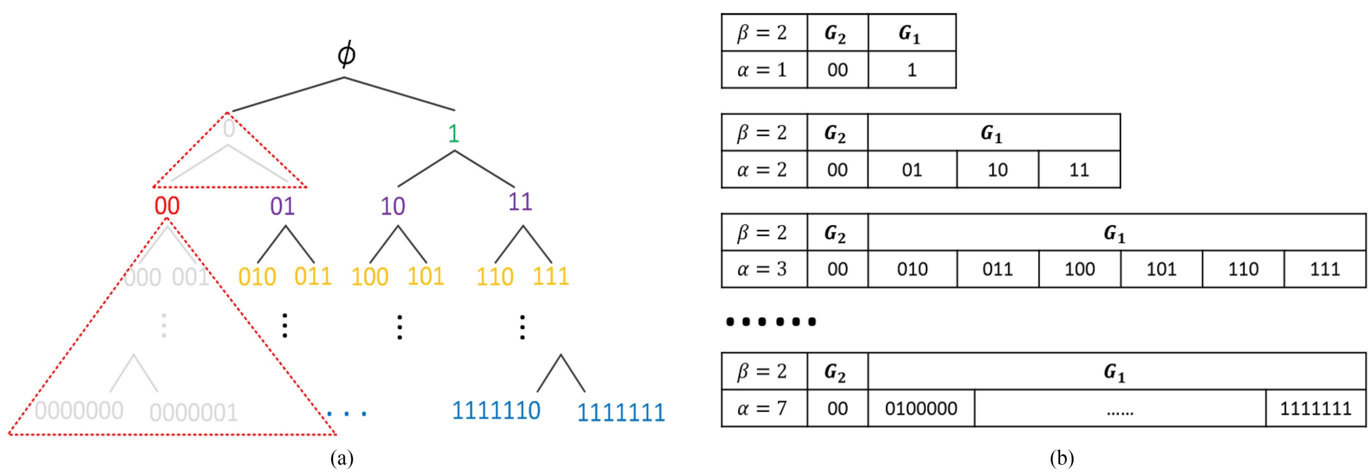


Fig. 3. Example of labeling bits selection when $\beta = 2$ and $\alpha = 1$ to 7.

II. Parametric Binary Tree Labeling Scheme

In this section, we propose a parametric binary tree labeling scheme (PBTL). It is designed to label pixels in two different categories, namely $\mathbf{G_1}$ and $\mathbf{G_2}$. For pixels with 8-bit depth, we use α and β bits of binary code to label pixels in $\mathbf{G_1}$ and $\mathbf{G_2}$, respectively, where $1 \leqslant \alpha, \beta \leqslant 7$.

To better explain our idea, we use a full binary tree structure, as shown in Fig. 1, to illustrate the distribution of binary labeling bits. As can be seen, the binary tree has 7 layers of child node, and the ith layer contains 2^i nodes, where $i = 1, 2, \ldots, 7$.

First of all, given a parameter β, we use the binary code in the first node of the βth layer to label pixels in $\mathbf{G_2}$. Thus, '$\underbrace{0\ldots0}_{\beta}$' is adopted. For $\mathbf{G_2}$, all pixels are labeled by the same labeling bits '$\underbrace{0\ldots0}_{\beta}$'. For $\mathbf{G_1}$, according to the known value β and another given parameter α, pixels are classified into n_a sub-categories, where n_α is calculated by Eq. (1).

$$n_\alpha = \begin{cases} 2^\alpha - 1, & \text{if } \alpha \leqslant \beta \\ (2^\beta - 1) * 2^{\alpha-\beta}, & \text{otherwise} \end{cases} \quad (1)$$

For pixels in a sub-category, we use the same α-bit binary code to label them, and for pixels in different sub-categories, different α-bit binary codes are applied. Thus, n_α different α-bit binary codes are utilized to label n_α sub-categories, respectively. Next, we analyze the content of these n_α binary codes from the following three aspects.

1) When $\alpha = \beta$, as shown in Fig. 1, the first node of the βth layer '$\underbrace{0\ldots0}_{\beta}$' is selected to label pixels in $\mathbf{G_2}$, and the remaining $n_\alpha = 2^\alpha - 1$ nodes of binary codes in the same layer are utilized to label pixels in n_α sub-categories of $\mathbf{G_1}$, respectively.
2) When $\alpha < \beta$, for each of the αth layer, we ignore the first node and use the remaining $n_\alpha = 2^\alpha - 1$ nodes of binary codes to label pixels in n_α sub-categories of $\mathbf{G_1}$. The illustrative examples can be found in Figs. 3 and 4.
3) When $\alpha > \beta$, for each of the αth layer, only the binary codes that are not derived from the node '$\underbrace{0\ldots0}_{\beta}$' are selected to label pixels in n_α sub-categories of $\mathbf{G_1}$. Figs. 2–4 show the examples of labeling bits selection when $\alpha = 1$ to 7, β equals to 1, 2 and 3, respectively. As can be seen, for example, in Fig. 2, all binary codes that derived from '0' are ignored and the remaining binary codes in αth layer are kept.

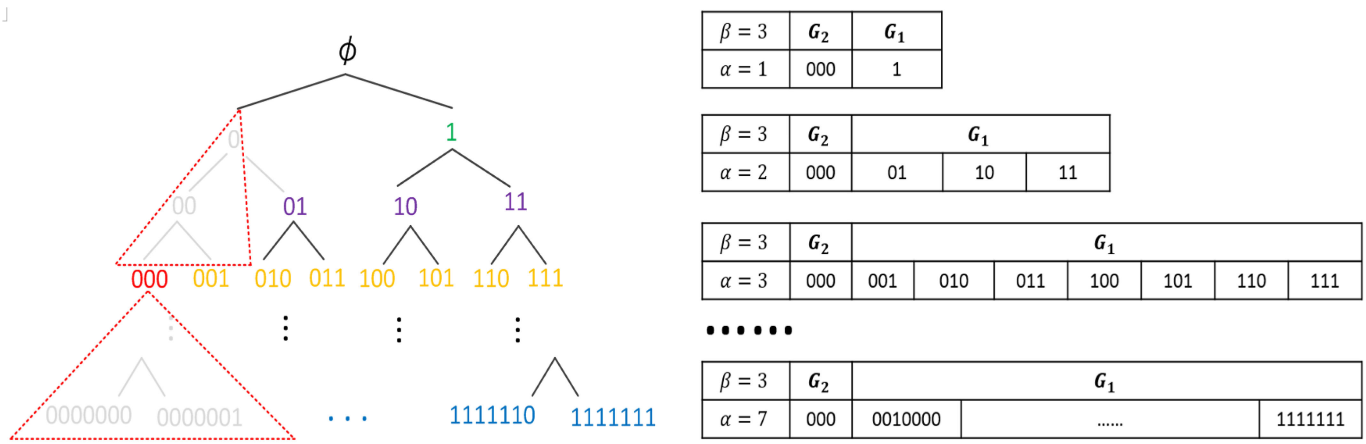


Fig. 4. Example of labeling bits selection when $\beta = 3$ and $\alpha = 1$ to 7.

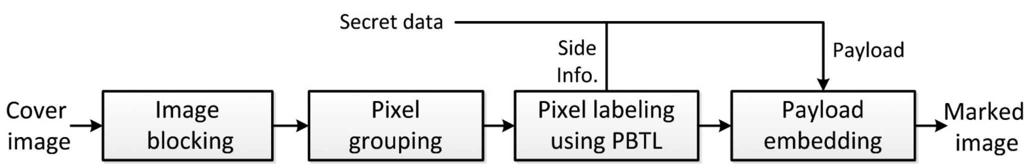


Fig. 5. Framework of PBTL–DE.

III. Data Embedding Using PBTL

Using PBTL, we propose a data embedding method (PBTL-DE) to embed secret data into a cover image. Firstly, we introduce how to embed secret data into a cover image using PBTL, and then explain how to extract secret data and recover the cover image.

A. Data Embedding

The framework of PBTL-DE is shown in Fig. 5. It consists four steps, namely *Step 1:* image blocking; *Step 2:* pixel grouping; *Step 3:* pixel labeling using PBTL and *Step 4:* payload embedding. Next, we introduce these four steps one by one.

Step 1: Image blocking: For an 8-bits depth original image $\mathbb{I}$ with a size of $M \times N$, we first divide it into a number of $s \times s$ non-overlapped small blocks. For example, the block size is set to 2×2 or 3×3.

Step 2: Pixel grouping: For all pixels in $\mathbb{I}$, we separate them into four sets, namely: reference pixel ($\mathbf{P_r}$), special pixel ($\mathbf{P_s}$), embeddable pixel ($\mathbf{P_e}$) and non-embeddable pixel ($\mathbf{P_n}$). Here, $\mathbf{P_r}$ consists of n_r pixels that selected by user-defined rules. For example, we select the first pixel (or center pixel) of each 2×2 (or 3×3) block to form $\mathbf{P_r}$. These pixels will be kept unmodified during data embedding phase. $\mathbf{P_s}$ contains one pixel which will be utilized to store some parameters. Any pixel except in $\mathbf{P_r}$ can be selected to be $\mathbf{P_s}$. Without loss of generality, we choose one pixel in the first block to be $\mathbf{P_s}$. Thus, for each block except for the first one, one reference pixel is corresponding to 3 (for 2×2) or 8 (for 3×3) non-reference pixels; otherwise, one reference pixel is corresponding to 2 (for 2×2) or 7 (for 3×3) non-reference pixels. Then, for each of the remaining $(MN - n_r - 1)$ pixels $\mathbb{I}_i (i = 1, 2, \ldots, MN - n_r - 1)$, we calculate its difference value e_i by

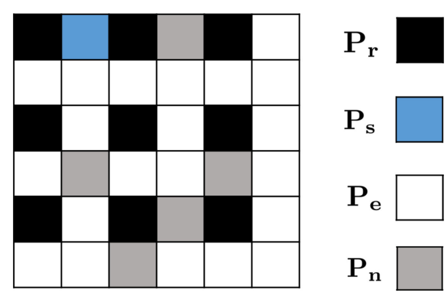


Fig. 6. Illustrative example of pixel grouping when block size is 2×2.

$$e_i = \mathbb{I}_i - \mathbb{I}_i^{\text{ref}} \tag{2}$$

where $\mathbb{I}_i^{\text{ref}} \in \mathbf{P_r}$ is the corresponding reference pixel of $\mathbb{I}_i$. If e_i satisfied the following condition, the pixel $\mathbb{I}_i$ belongs to $\mathbf{P_e}$; otherwise, it is in set $\mathbf{P_n}$.

$$\left\lceil -\frac{n_\alpha}{2} \right\rceil \leqslant e_i \leqslant \left\lfloor \frac{n_\alpha - 1}{2} \right\rfloor \tag{3}$$

where n_a is a positive integer, $\lceil * \rceil$ and $\lfloor * \rfloor$ are the ceil and floor operations, respectively. Here, $\mathbf{P_e}$ and $\mathbf{P_n}$ contain n_e and n_n pixels, respectively, where pixels in $\mathbf{P_e}$ can be utilized to embed secret data while $\mathbf{P_n}$ can not. Thus, $MN = n_r + n_e + n_n + 1$. Fig. 6 shows an example of pixel grouping when block size is 2×2.

After obtaining the difference set $\mathbf{e} = \{e_i\}_{i=1}^{MN-n_r-1}$, we can obtain its histogram $h(e)$ by

$$h(e) = \#\{1 \leqslant i \leqslant MN - n_r - 1 : e_i = e\}, \quad \forall e \in \mathbf{Z} \tag{4}$$

Algorithm 1: PBTL-DE.

Input: Original image $\mathbb{I}$, Secret data $\mathcal{M}$, parameters α and β.

```
Divide 𝕀 into equal size non-overlapping blocks and classify pixels into four groups P_r, P_s, P_e and P_n.
Construct the payload 𝒫, where it consists of the secret data ℳ, the first β bits of each pixel in P_n and 8 bits in P_s.
for each pixel in P_e do
    According to the difference value e, reconstruct the pixel by replacing α labeling bits and (8 − α) payload bits.
end for
for each pixel in P_n do
    Replace its first β bits by '0...0' (β) and keep the remain (8 − β) bits unmodified.
end for
Convert α and β into binary bits and store them into P_s by bit replacement.
```

Output: Marked image $\hat{\mathbb{I}}$.

where $\#$ is the cardinal number of a set. Due to the spatial correlations of pixels within the same block, the histograms of $\mathbf{e}$ form like a Laplace distribution with location parameter equals to 0. As shown in Eq. (3), in order to achieve a higher embedding rate, we use the pixel whose difference value falls into the n_α center bins of histogram $h(e)$ to embed secret data.

Step 3: Pixel labeling using PBTL: Because the pixel locations of $\mathbf{P_r}$ and $\mathbf{P_s}$ are pre-defined, they can be easily distinguished, we only need to label the pixels in $\mathbf{P_n}$ and $\mathbf{P_e}$. Given two parameters α and β, we use the binary codes generated by PBTL to label pixels in $\mathbf{P_n}$ and $\mathbf{P_e}$, respectively. For example, for each pixel in $\mathbf{P_n}$, a β-bit binary code '$\underbrace{0\ldots0}_{\beta}$' is adopted to label it by bit replacement, and the remaining $(8-\beta)$ bits are kept unmodified. For pixels in $\mathbf{P_e}$, they can be classified into n_α sub-categories according to different values of e. Thus, n_α different α-bit binary codes are utilized to label pixels in each sub-category, respectively.

Step 4: Payload embedding: The payload contains three parts: the original 8 bits of pixel in $\mathbf{P_s}$, the replaced original β bits of each pixel in $\mathbf{P_n}$, and the secret data.

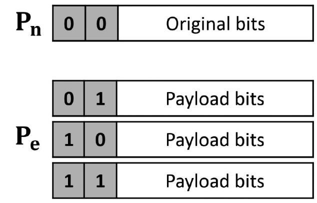
After pixel labeling, the remaining $(8-\alpha)$ bits of each pixel in $\mathbf{P_e}$ are reserved to embed payload bits by bit replacement. Thus, totally $(8-\alpha)n_e$ bits of the payload can be successfully embedded. The parameters α and β are important for data extraction and image recovery, thus, they need to be stored as well. Since $1 \leqslant \alpha, \beta \leqslant 7$, they can be successfully stored by 8 bits in $\mathbf{P_s}$ by bit replacement. Therefore, the marked image is generated, and the detailed procedures of RDH using PBTL are provided in Algorithm 1.

Then, we can calculate the effective embedding rate $r_{\alpha,\beta}$ (.bpp) under different settings of parameters α and β by

$$r_{\alpha,\beta} = \frac{(8-\alpha)n_e - \beta n_n - 8}{MN} \tag{5}$$

$\mathbf{P_n}$	$\mathbf{P_e}$		
00	$e=-1$	$e=0$	$e=1$
	01	10	11

(a)



(b)

Fig. 7. Illustrative example of pixel labeling and data embedding when $\alpha = \beta = 2$. (a) Binary code distribution. (b) Pixel bits in $\mathbf{P_e}$ and $\mathbf{P_n}$ after data embedding.

which is equivalent to

$$r_{\alpha,\beta} = \frac{(8-\alpha)\sum_{i=v_l}^{v_r} h(i) - \beta\left(\sum_{j=-255}^{v_l-1} h(j) + \sum_{k=v_r+1}^{255} h(k)\right) - 8}{MN} \tag{6}$$

where $v_l = \lceil -\frac{n_\alpha}{2} \rceil$ and $v_r = \lfloor \frac{n_\alpha - 1}{2} \rfloor$. Then the maximum embedding rate $r_{\max}$ (.bpp) can be calculated by

$$r_{\max} = \max\{r_{\alpha,\beta}\}_{\alpha,\beta=1}^{7} \tag{7}$$

An illustrative example of pixel labeling and data embedding when $\alpha = \beta = 2$ is shown in Fig. 7. As can be seen, '00' is utilized to label pixels in $\mathbf{P_n}$, and the remaining 6 bits are kept unmodified. According to Eq. (1), $n_\alpha = 3$. Thus, '01', '10' and '11' are applied to label pixels in $\mathbf{P_e}$ when the difference value e equal to -1, 0 and 1, respectively.

B. Data Extraction and Image Recovery

After obtaining the marked image, according to the parameters extracted from $\mathbf{P_s}$, we check the labeling bits of non-reference pixels and classify them into sets $\mathbf{P_e}$ and $\mathbf{P_n}$, accordingly. When the first β bits equal to '$\underbrace{0\ldots0}_{\beta}$', the pixel is grouped in $\mathbf{P_n}$; otherwise, it is put in $\mathbf{P_e}$. We then extract $(8-\alpha)$ bits of the payload from the pixel in $\mathbf{P_e}$ sequentially. Next, for each pixel in $\mathbf{P_e}$, according to its α-bit labeling bits, we obtain the corresponding difference value e and recover the pixel by

$$\mathbb{I}_i = \mathbb{I}_i^{\text{ref}} + e_i \tag{8}$$

Finally, we recover the replaced two bits of each pixel in $\mathbf{P_n}$ and 8 bits of the pixel in $\mathbf{P_s}$ using the extracted payload. By now the secret data is successfully extracted and the original image is fully recovered.

IV. Reversible Data Hiding in Encrypted Images Using PBTL

Using PBTL-DE, we propose a RDH method in encrypted images (PBTL-RDHEI). It encrypts the original image into a noise like one while keeping spatial correlations within small image blocks, so that PBTL-DE can be applied to embed secret data into the encrypted image. PBTL-RDHEI consists of three phases as shown in Fig. 8, namely A) Generation of encrypted image; B) Generation of marked encrypted image and

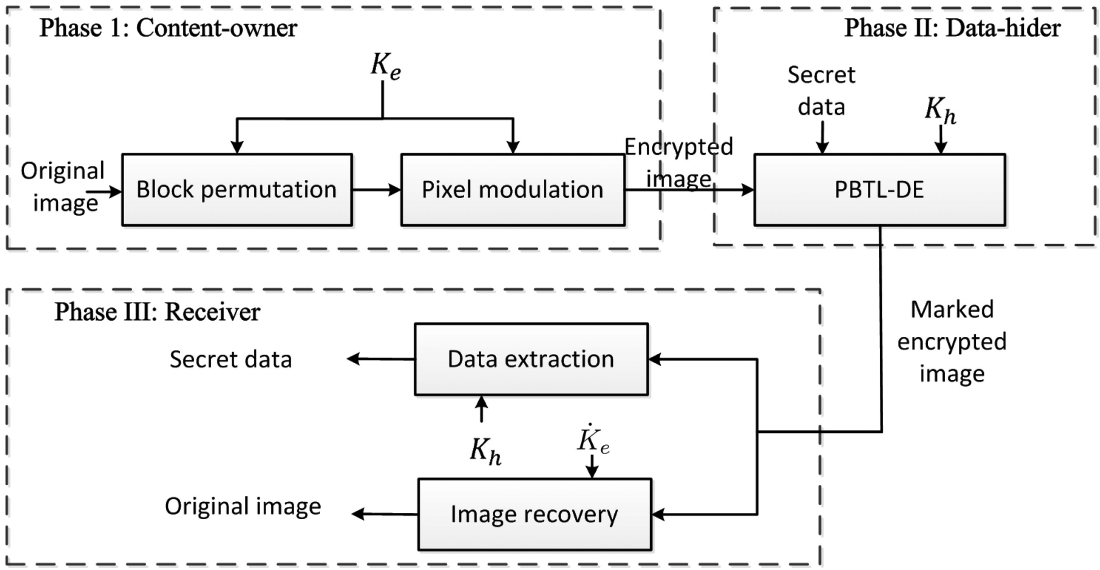


Fig. 8. Framework of PBTL–RDHEI.

C) Data extraction and image recovery. These three phases are accomplished by the content-owner, data-hider and receiver, respectively. At the receiver side, using different security keys, the secret data and original image can be successfully extracted and recovered. Next, we will introduce these three phases one by one.

A. *Generation of Encrypted Image*

Image encryption consists of two procedures: block permutation and pixel modulation. Assume that an 8-bit depth gray-scale image $\mathbb{O}$ is with a size of $M \times N$. Firstly, the content-owner divides $\mathbb{O}$ into k non-overlapped blocks $\mathbb{O}_{(i)} (i = 1, 2, \ldots, k)$ with a size of $s \times s$, where $k = MN/s^2$, and s is a small integer that greater than or equal to 2. Then, image blocks are permuted according to $\dot{K}_e$, where $\dot{K}_e = \mathcal{H}(\mathbb{O}) \oplus \mathcal{H}(K_e), \mathcal{H}(.)$ is a secure hash function to produce a hash sequence, '$\oplus$' is the bit-level XOR operation. Users have flexibility to select any pixel permutation method for image scrambling, and we consider the image block as a single unit. Here we use the scrambling method in [37] for demonstration. The scrambled image blocks are denoted as $\hat{\mathbb{O}}_{(i)} (i = 1, 2, \ldots, k)$. Next, pixels in $\hat{\mathbb{O}}_{(i)}$ are modified by

$$\mathbb{E}^j_{(i)} = (\hat{\mathbb{O}}^j_{(i)} + R_i) \mod 256, \qquad (j = 1, 2, \ldots, s^2) \tag{9}$$

where $\hat{\mathbb{O}}^j_{(i)}$ is the jth pixel of block $\hat{\mathbb{O}}_{(i)}$ in raster-scan order, $R_i \in [0, 255]$ is a random integer generated by $\dot{K}_e$. Any random number generator can be used to generate R_i. As an example, we use $\dot{K}_e$ to initialize (r, x_0) of the Tent-Sine system (TSS) [38] to obtain the random number for demonstration, where the TSS is defined by

$$x_{i+1} = \begin{cases} rx_i/2 + (4 - r)\sin(\pi x_i)/4 \mod 1, & x_i < 0.5 \\ r(1 - x_i)/2 + (4 - r)\sin(\pi x_i)/4 \mod 1, & x_i \geqslant 0.5 \end{cases} \tag{10}$$

Algorithm 2: Generation of initial condition of TSS

Input: $\dot{K}_e = [k_1, k_2, \ldots, k_{256}](k_i \in \{0, 1\}, 1 \leqslant i \leqslant 256)$.
$u_1 \leftarrow \sum_{i=1}^{64} k_i 2^{64-i}$
$u_2 \leftarrow \sum_{i=64}^{128} k_i 2^{128-i}$
$v_1 \leftarrow \sum_{i=129}^{192} k_i 2^{192-i}$
$v_2 \leftarrow \sum_{i=193}^{256} k_i 2^{256-i}$
Initial value $x_0 \leftarrow u_1/2^{90}$
Parameter $r_0 \leftarrow u_2/2^{90}$
for $i = 1$ to 2 **do**
  $x_i \leftarrow (x_{i-1} u_i v_i / 2^{80} + x_{i-1}) \mod 1$
  $r_i \leftarrow (r_{i-1} u_i v_i / 2^{80} + r_{i-1}) \mod 4$
end for
$r \leftarrow 4 - r_2$
$x_0 \leftarrow x_2$
Output: Initial conditions (r, x_0).

where (r, x_0) is the initial condition, $r \in (0, 4]$ and $x_i \in (0, 1)$. Here, we use SHA-256 for demonstration. It is sensitive to the input content and produces a 256-bit binary hash sequence. The user-defined key K_e also contains 256 bits. The obtained $\dot{K}_e$ is applied to initialize (r, x_0) using Algorithm. 2. The random value R_i is calculated by

$$R_i = \lfloor x_{i+1} * 2^{40} \mod 256 \rfloor \tag{11}$$

According to Eq. (9), the s^2 pixels in the ith image block are added by a same random integer R_i for modulation. Thus, spatial correlations will be kept in the image blocks of $\mathbb{D}$, and they can be exploited to embed secret data at the data-hider side. However, the block size will influence the encryption effect. To keep a relatively high security, in this study, we set the block size to 2×2 and 3×3 for demonstration.

B. Generation of Marked Encrypted Image

After obtaining the encrypted image $\mathbb{E}$, the data-hider first divides it into k blocks by the same way in image encryption phase. Given the parameters α and β, we then can embed payload bits into image $\mathbb{E}$ using PBTL-DE as provided in Algorithm 1. In order to achieve a higher security, the secret data is encrypted using K_h before embedded into the encrypted image. We denote the image $\mathbb{E}$ after data embedded in as the marked encrypted image $\mathbb{M}$.

C. Data Extraction and Image Recovery

At the receiver side, authorized users with different security keys are able to obtain different contents, secret data, original image or both, separately.

1) Data Extraction: When holding the data hiding key K_h, the receiver can extract the secret data successfully. Firstly, we divide $\hat{\mathbb{M}}$ into k blocks as in data embedding phase, extract parameters α and β from $\mathbf{P_s}$. We keep the reference pixels in $\mathbf{P_r}$ unmodified. For the rest $3 * k - 1$ pixels, by checking the first α and β labeling bits in each pixel, we classified them into $\mathbf{P_e}$ and $\mathbf{P_n}$. Then, we extract $(8 - \alpha)$ bits of payload from pixels in $\mathbf{P_e}$ sequentially and obtain the encrypted secret data from the extracted payload bits. Finally, using K_h, we decrypt the encrypted secret data to obtain the plaintext secret data.

2) Image Recovery: Using $\dot{K}_e$, the receiver is able to obtain a recovered image that is exactly the same with the original one. Firstly, after obtaining the payload from $\hat{\mathbb{M}}$ as in data extraction phase, we recover the replaced β bits in each pixel of $\mathbf{P_n}$ and 8 bits in $\mathbf{P_s}$ using the payload. Thus, all pixels except in $\mathbf{P_e}$ are recovered. For each pixel in $\mathbf{P_e}$, according to its α labeling bits, we find its corresponding difference value e and recover it using Eq. (8). By now the obtained image $\mathbb{E}$ is the same as before data embedding. Next, we recover pixel values in $\mathbb{E}$ by

$$\hat{\mathbb{O}}^{j}_{(i)} = (\mathbb{E}^{j}_{(i)} - R_i) \mod 256, \quad (j = 1, 2, \ldots, s^2) \tag{12}$$

where R_i is generated by the same way in image encryption phase. Finally, we inversely permute the image blocks in $\hat{\mathbb{O}}$ and obtain the original image $\mathbb{O}$.

Due to the reversibility in each step of data extraction and image recovery, the secret data and original image are obtained without any error.

V. Discussion

In PBTL-RDHEI, the encryption process uses block permutation and modulation to transform the original image into a noise-like one while keeping spatial redundancy within small image blocks. Data hiding is then applied to the encrypted image by exploiting the spatial redundancy within the small encrypted image blocks. The proposed image encryption method modulates each pixel within an image block using the same random integer (see Eq. (9)). The encrypted image block will retain the spatial redundancy as it is in the original image. Therefore, images with less texture in the original image have higher spatial redundancy and thus can be embedded with more data, achieving a larger embedding rate. However, the modulation operation may also reduce the spatial correlations of pixels within the block. Here, we analyze the redundancy preserving of the image encryption method in PBTL-RDHEI.

Take the block size 2×2 for example, we denote the 4 pixels in an original image block X as x_1, x_2, x_3 and x_4, where x_1 is the reference pixel and $x_i \in [0, 255]$ (for $i = 1, 2, 3, 4$). Then we can obtain their difference values $e_j = x_1 - x_j$ (for $j = 2, 3, 4$). After using the pixel modulation operation with a random value v ($v \in [0, 255]$), the four pixels $\hat{x}_i$ in the new image block $\hat{X}$ are calculated by $\hat{x}_i = (x_i + v) \mod 256$, and the new difference values $\hat{e}_j$ are obtained using $\hat{e}_j = \hat{x}_1 - \hat{x}_j$. If $\hat{e}_j = e_j$ for $j = 2, 3, 4$, block $\hat{X}$ and X have the same spatial correlation. To meet this requirement, one of the following two conditions needs to be satisfied.

$$v + x_{\max} < 256 \tag{13}$$

or

$$v + x_{\min} \geqslant 256 \tag{14}$$

where $x_{\max} = \max\{x_2, x_3, x_4\}$ and $x_{\min} = \min\{x_2, x_3, x_4\}$.

For example, assume that $X = [102, 102; 99, 103]$, we obtain $e_2 = 0$, $e_3 = 3$, $e_4 = -1$, $x_{\max} = 103$ and $x_{\min} = 99$. If $v = 100$, $v + x_{\max} = 203 < 256$, Eq. (13) is satisfied, and $\hat{X} = [202, 202; 199, 203]$. Thus, $\hat{e}_2 = e_2 = 0$, $\hat{e}_3 = e_3 = 3$, $\hat{e}_4 = e_4 = -1$. If $v = 153$, non of the conditions in Eq. (13) and (14) is satisfied, and $\hat{X} = [255, 255; 252, 0]$. Thus, $\hat{e}_2 = e_2 = 0$, $\hat{e}_3 = e_3 = 3$, $\hat{e}_4 = 255 \neq e_4$.

In order to analyze the inference of spatial correlations in block cause by using the pixel modulation operation, we calculate the histogram of difference value set $\mathbf{e}$ of *Lena* image with or without modulation operation. The results are plotted in Fig. 9. From the results, we can observe that, the histogram almost keep the same shape. This means that, by using the block based pixel modulation operation, the spatial correlation of pixels within the same block is well kept, which ensuring the high embedding rate of the proposed PBTL-RDHEI.

VI. Security and Performance Analysis

In this section, we analyze the security of PBTL-RDHEI, and compare encryption performance and efficiency between PBTL-RDHEI and several state-of-the-art methods.

A. Security Analysis

As an encryption domain based RDH method, the PBTL-RDHEI should ensure the security to both the original image and secret data. In PBTL-RDHEI, the security to secret data is provided by data encryption with K_h. It is worth noting that any secure data encryption algorithm can be used here to encrypt the secret data. Hence, without K_h, it is extremely difficult to reveal the secret data. Therefore, we mainly focus on analyzing security of the image encryption algorithm in withstanding serval attacks such as the brute-force attack, known/chosen-plaintext attack and noise attack.

1) Brute-Force Attack: The image encryption process including block permutation and modulation. For an $M \times N$ original image with block size $s \times s$, there are $(MN/s^2)!$ possible

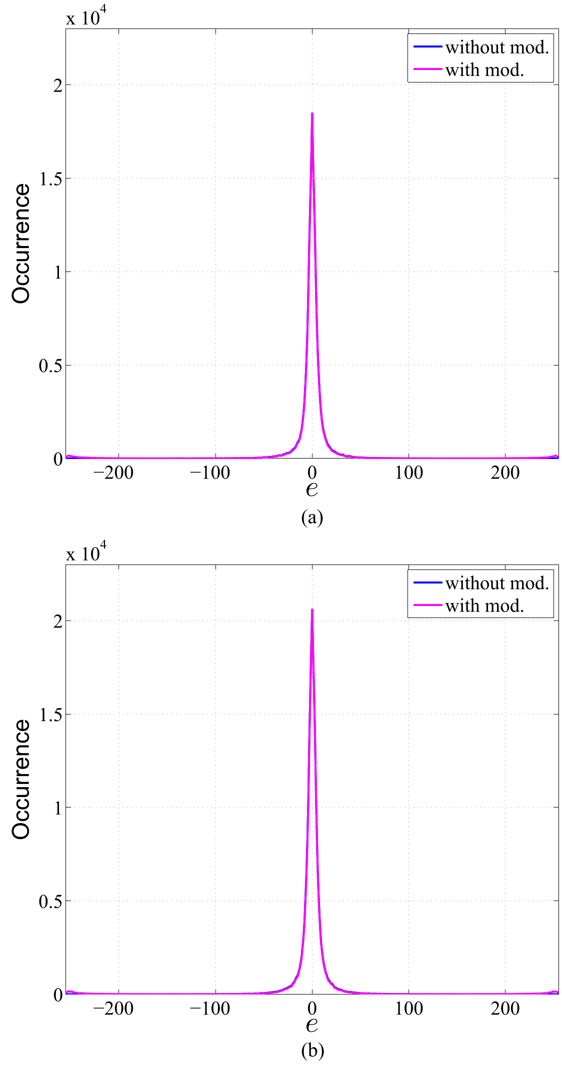


Fig. 9. Histograms of difference e calculated from image $Lena$ when applied with and without pixel modulation operation for block sizes are set to (a) 2×2 and (b) 3×3.

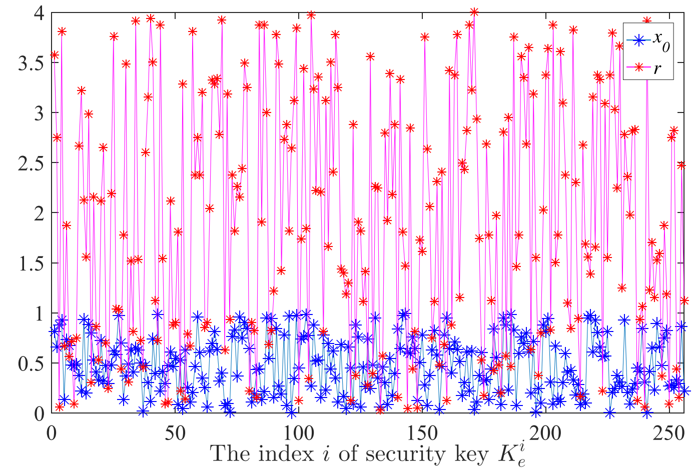


Fig. 10. Distribution of initial condition (r, x_0) generated by 257 different keys.

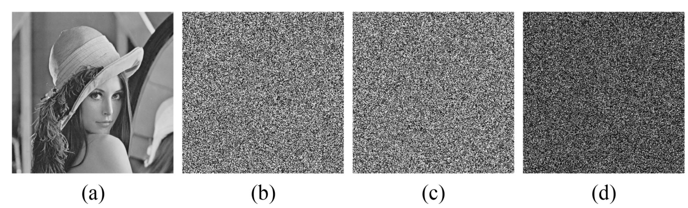


Fig. 11. Simulation results of encrypted *Lena* images using two keys K_e^1 and K_e^2 when block size is set to 2×2, where K_e^1 and K_e^2 are with only one bit difference. (a) The original image. (b) Encrypted image using K_e^1. (c) Encrypted image using K_e^2. (d) The difference between (a) and (d).

permutations. For pixel modulation, each value of $R_i \in [0, 255]$ is randomly generated. Thus, the possibility of successfully obtain the original image by data-hider without K_e is as small as $\frac{1}{(MN/s^2)!256^{MN/s^2}}$, so that the security can be ensured. As can be seen, with large image size and small block size, a higher encryption performance to the original image can be achieved.

In order to analyze the key sensitivity in withstanding the brute-force attack, we use a user-defined 256-bits security key K_e and other 256 keys K_e^i $(i = 1, 2, \ldots, 256)$ to generate initial conditions (r, x_0) for TSS, where K_e^i is the same as K_e except for flipping the ith bit in K_e. Therefore, K_e^i and K_e are of only one bit difference. Because TSS is sensitive to the initial condition, different settings of (r, x_0) will result in totally different random values R_i and thus different encrypted images. Fig. 10 shows the results of initial condition (r, x_0) generated by these 257 keys. As can be seen, tinny changes in the key will significantly change the initial condition of TSS. We then show the simulation results of encrypting *Lena* image using two keys and obtain the difference of these two encrypted images, where the two keys are with only one bit difference. The results are shown in Fig. 11. As can be seen, even with one bit change in the encryption keys, the obtained encrypted images are totally different. Therefore, the attacker has extreme difficulty in revealing the original image by analyzing the security key. The proposed algorithm is able to withstand brute-force attack.

2) Known/Chosen-Plaintext Attack: Known-plaintext attack is a cryptanalysis model that the attackers have the plaintexts and their corresponding ciphertexts and try to reveal all or part of the secret key. A chosen-plaintext attack is more powerful than known-plaintext attack, because the attackers can arbitrarily choose plaintexts for encryption and obtain the corresponding ciphertexts. Therefore, an encryption algorithm that can withstand chosen-plaintext attack is also secure against known-plaintext attack.

In order to show the robustness of the proposed encryption algorithm in withstanding chosen-plaintext attack, we perform differential analysis to the proposed algorithm. Two original images are encrypted using the same key and obtain the difference between their cipher images. Here the two original images are of one bit difference. The simulation results of differential analysis are shown in Fig. 12. From the results, we can observe that a slight change in the original image will result in a totally different encrypted image, which is extremely difficult for attackers to obtain useful information by analyzing the pairs of plaintexts and ciphertexts. Therefore, our algorithm can withstand the known/chosen-plaintext attack.

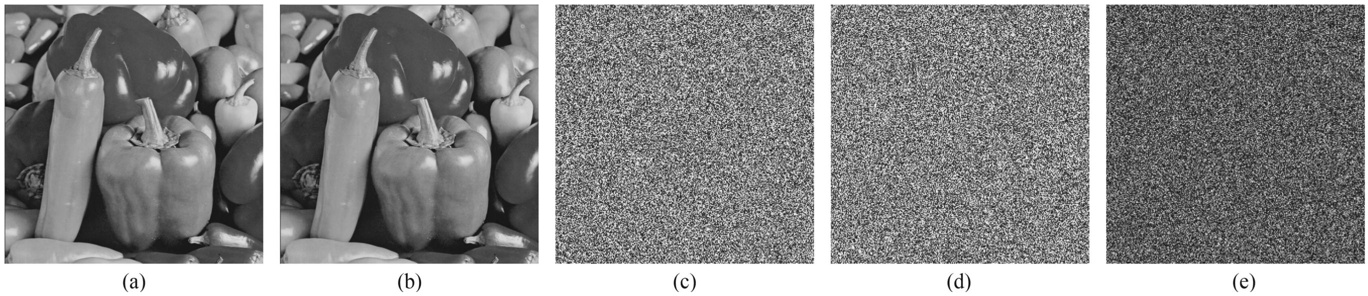


Fig. 12. Simulation results of differential analysis on *Peppers* image when block size is set to 2×2. (a) The original image $\mathbb{O}_1$. (b) The original image $\mathbb{O}_2$, where $\mathbb{O}_1$ and $\mathbb{O}_2$ are with only one bit difference. (c) Encrypted results of $\mathbb{O}_1$. (d) Encrypted results of $\mathbb{O}_2$. (e) The difference between (c) and (d).

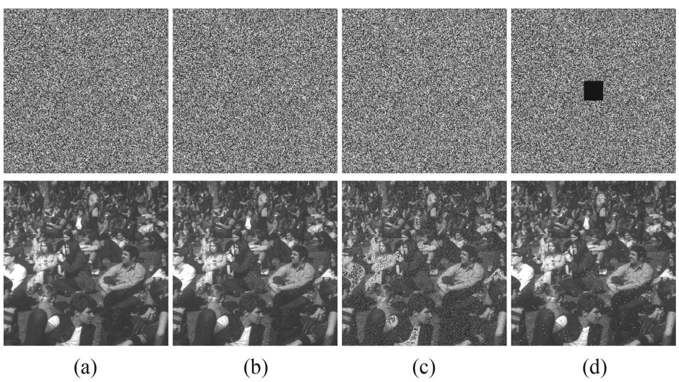


Fig. 13. Noise attack analysis of the encryption method in PBTL–RDHEI when block size is 2×2. The top row shows the encrypted *Crowd* images (a) without noise; (b) with 1% salt and pepper noise; (c) with 1% Gaussian noise; (d) with 60×60 image cut, respectively. The bottom row shows the recovered images.

3) Noise and Data Loss Attacks: Fig. 13 shows the noise and data loss attacks to the encrypted images. We use the *Crowd* image for demonstration and set the block size to 2×2. The results show that most of the original image information can be recovered when the encrypted image is added with 1% of the noise (Salt & Pepper noise or Gaussian noise) or with 60×60 pixel cutting.

B. Performance Analysis

Here, we compare the encryption performance and efficiency between the proposed algorithm and several related works.

1) Encryption Performance Comparison: Table I compares the image encryption performance of the proposed algorithm with that of several state-of-the-art methods. As can be seen, the proposed encryption algorithm makes the histogram of the encrypted image uniform distributed. Although it preserves spatial redundancy within small image blocks like the methods in [24] and [22], it has high security level that can withstand chosen-plaintext and other attacks.

2) Efficiency Analysis: Time and space complexities are often used to estimate the efficiency of an algorithm. Time complexity, also called the computation complexity, is to measure the running time of an algorithm. Space complexity is a measure of working storage that an algorithm needs with respect to the input content. In PBTL-RDHEI, the encryption/decryption process contains block permutation and modulation. The processing time doesn't rely on the content of the original image. Given an image with k blocks, the time and space complexities of image encryption/decryption are $O(k)$ and $O(MN + k)$, respectively. The data embedding/extraction mainly contains pixel grouping and modification. Each one needs to go through all pixels only once except for $\mathbf{P_r}$. Therefore, the time complexity of data embedding/extraction is $O(MN - k)$. In the embedding/extraction process, a temporary workspace is needed to store the original bits in $\mathbf{P_n}$ and $\mathbf{P_s}$ that are replaced by labeling bits and parameters. The temporary workspace is less than MN. Therefore, given the secret data with size of N_d, the space complexity of data embedding/extraction is $O(MN + N_d)$.

Next, we experimentally compare the time complexity of PBTL-RDHEI with that of several related works under various embedding rates. The measure of the time complexity is carried out over the Matlab implementation by using the built-in time function in a workstation with Intel i7@3.40 GHz CPU and 8 GB RAM. We randomly selected 100 images with size of 512×512 from the BowsBase[1] database for testing, the average results of time complexity under various embedding rates are plotted in Fig. 14. Here the time is recorded for the processes of image encryption, data embedding, data extraction and image recovery. The images with sizes of 128×128 and 256×256 are generated by downsampling the selected 100 images. We can observe that images with a larger size will require more time to finish all processes, and that the proposed PBTL-RDHEI has almost the lowest computation cost under various embedding rates. This verifies the efficiency of PBTL-RDHEI.

VII. Experiment Results and Comparisons

In this section, we show the experiment results and comparisons of PBTL-RDHEI with several existing related works. Three image database are used in this section, including the Miscelaneous,[2] Kodak[3] and BowsBase. Four commonly used test images *Lena*, *Airplane*, *Man* and *Crowd* are selected from the Miscelaneous database, two images *kodim08* and *kodim13* are selected from the Kodak database and the rest test images are selected from the BowsBase database.

Fig. 16 shows the distribution of pixel values of $Lena$ image after image encryption and data embedding processes,

[1]http://bows2.ec-lille.fr/
[2]http://decsai.ugr.es/cvg/dbimagenes/g512.php
[3]http://www.r0k.us/graphics/kadak

TABLE I
ENCRYPTION PERFORMANCE COMPARISON OF THE PROPOSED ALGORITHM WITH SEVERAL RELATED WORKS

	Encryption method	Uniformed histogram in encrypted image	Redundancy preserving in small image blocks	Chosen-plaintext attack	Noise attack
Zhang [16]	Stream cipher	Yes	No	No	Yes
Yin [22]	Block permutation	No	Yes	No	Yes
Zhou [9]	Stream cipher	Yes	No	No	Yes
Ma [11]	Stream cipher	Yes	No	No	No
Cao [18]	Stream cipher	Yes	No	No	No
Huang [24]	Block permutation, Stream cipher	Yes	Yes	No	Yes
Proposed	Block permutation, Block modulation	Yes	Yes	Yes	Yes

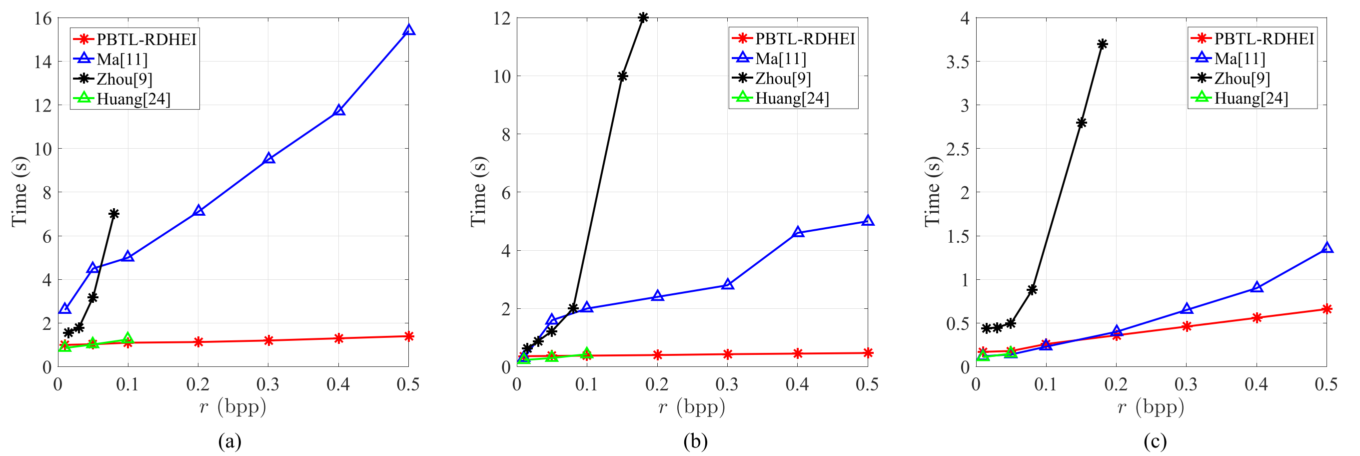
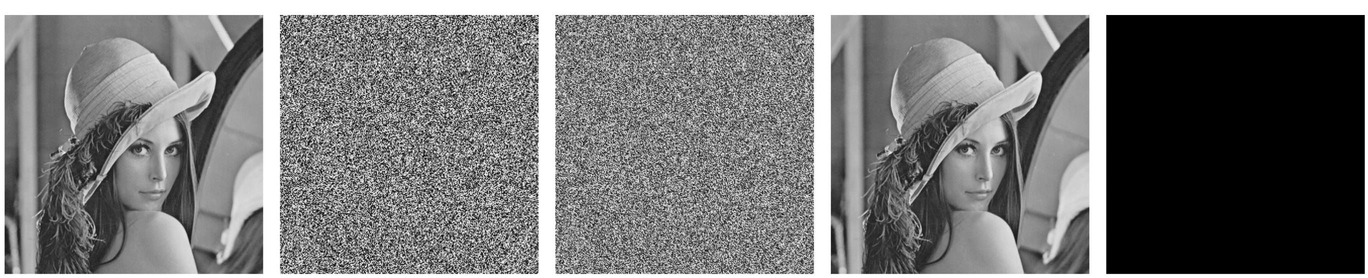


Fig. 14. Time complexity of PBTL–RDHEI and several related works under various embedding rates and image sizes. (a) 512×512. (b) 256×256. (c) 128×128.



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

Fig. 15. Simulation results of applying PBTL–RDHEI to *Lena* image when block size is set to 2×2. (a) The original image. (b) Encrypted image. (c) Marked encrypted image with $\alpha = 5$ and $\beta = 2$, $r_{\max} = 1.722$ bpp. (d) recovered image, PSNR $= +\infty$ dB. (e) The difference between (a) and (d).

respectively. From the results, we can observe that, pixel values of the encrypted image and marked encrypted image are uniform distributed. Although the data hider is able to know the pixel spatial correlations within a block, s/he can not obtain the relationship between blocks, since the block is small, and block locations and pixel values within the block are changed.

Fig. 15 shows the experiment results of applying PBTL-RDHEI to *Lena* image when block size is set to 2×2, parameters $\alpha = 5$ and $\beta = 2$. From the results, we can observe that under the given settings of block size and parameters, the proposed algorithm reaches the maximum embedding rate of 1.722 bpp. Due to the reversibility of PBTL-RDHEI, the original image can be successfully recovered without any error (see Fig. 15(d)).

Tables II–V show the embedding rates of test images $Lena$, $Airplane$, Man and $Crowd$ under various α and β when block size is set to 2×2 and 3×3, respectively. From the results, we can observe that when α is set to small values, e.g., $\alpha = 1$ or 2, the marked encrypted image is unable to or can only embed a few secret data. For different images, the parameters set to achieve the maximum embedding rate are different. In general, images with block size 3×3 are able to embed more secret data than that of 2×2, because more pixels will be classified into the embeddable pixel category. In addition, images with less texture in the original version can achieve larger maximum embedding rate. For example, image $Airplane$ can reach the maximum embedding rate of 1.903 bpp and 2.188 bpp when block size is set to 2×2 and 3×3, respectively.

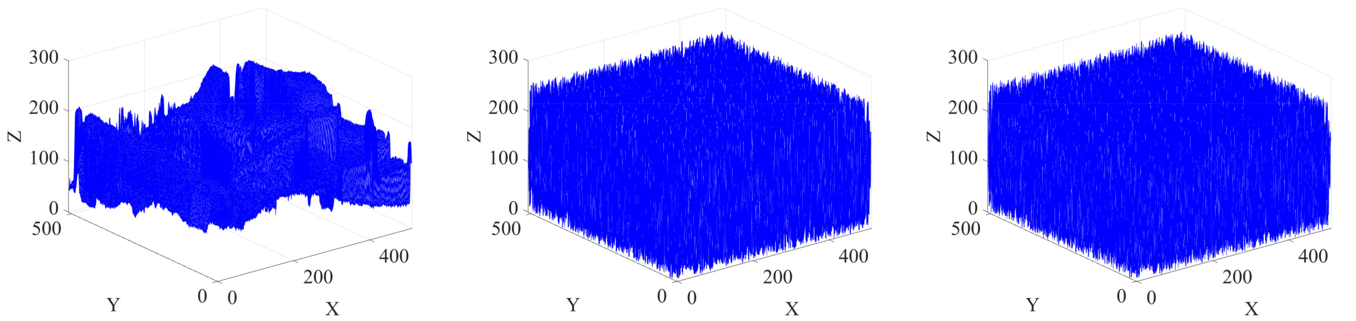


Fig. 16. Distribution of pixel values of $Lena$ image in each step when block size is 2×2. (a) The original image. (b) The encrypted image. (c) The marked encrypted image.

TABLE II
EMBEDDING RATES OF IMAGE $Lena$ UNDER VARIOUS α AND β WHEN BLOCK SIZE IS 2×2 AND 3×3

$r_{\alpha,\beta}$ (bpp)		β (block size 2x2)							β (block size 3x3)						
		1	2	3	4	5	6	7	1	2	3	4	5	6	7
α	1	-0.186	-0.866	-1.545	-2.225	-2.904	-3.584	-4.264	-0.253	-1.057	-1.860	-2.664	-3.467	-4.270	-5.074
	2	0.159	0.022	-0.537	-1.097	-1.657	-2.216	-2.776	0.155	-0.023	-0.687	-1.352	-2.016	-2.680	-3.345
	3	0.702	0.861	0.800	0.431	0.063	-0.306	-0.675	0.787	0.963	0.883	0.442	0.001	-0.440	-0.881
	4	1.324	1.619	1.635	1.568	1.390	1.211	1.032	1.520	1.874	1.894	1.807	1.592	1.377	1.162
	5	1.579	**1.722**	1.717	1.677	1.621	1.543	1.464	1.842	**2.018**	2.016	1.966	1.900	1.807	1.713
	6	1.273	1.308	1.299	1.277	1.249	1.218	1.183	1.494	1.539	1.528	1.502	1.469	1.433	1.392
	7	0.681	0.679	0.664	0.646	0.627	0.607	0.587	0.801	0.797	0.780	0.758	0.736	0.712	0.689

TABLE III
EMBEDDING RATES OF IMAGE $Airplane$ UNDER VARIOUS α AND β WHEN BLOCK SIZE IS 2×2 AND 3×3

$r_{\alpha,\beta}$ (bpp)		β (block size 2x2)							β (block size 3x3)						
		1	2	3	4	5	6	7	1	2	3	4	5	6	7
α	1	0.142	-0.497	-1.135	-1.774	-2.412	-3.051	-3.689	0.114	-0.644	-1.401	-2.159	-2.916	-3.674	-4.431
	2	0.624	0.759	0.292	-0.176	-0.643	-1.111	-1.579	0.673	0.804	0.243	-0.318	-0.879	-1.440	-2.002
	3	1.275	1.512	1.508	1.228	0.947	0.667	0.387	1.433	1.707	1.679	1.337	0.996	0.655	0.313
	4	1.722	**1.903**	1.876	1.799	1.649	1.499	1.349	1.973	**2.188**	2.151	2.047	1.862	1.677	1.492
	5	1.678	1.742	1.716	1.662	1.600	1.519	1.437	1.941	2.023	1.992	1.926	1.847	1.748	1.648
	6	1.262	1.277	1.258	1.227	1.190	1.152	1.109	1.473	1.492	1.468	1.431	1.387	1.339	1.286
	7	0.664	0.660	0.643	0.623	0.601	0.578	0.554	0.777	0.774	0.755	0.732	0.706	0.679	0.652

TABLE IV
EMBEDDING RATES OF IMAGE Man UNDER VARIOUS α AND β WHEN BLOCK SIZE IS 2×2 AND 3×3

$r_{\alpha,\beta}$ (bpp)		β (block size 2x2)							β (block size 3x3)						
		1	2	3	4	5	6	7	1	2	3	4	5	6	7
α	1	-0.290	-0.983	-1.675	-2.368	-3.061	-3.753	-4.446	-0.361	-1.178	-1.994	-2.811	-3.628	-4.445	-5.262
	2	-0.052	-0.368	-0.977	-1.585	-2.194	-2.802	-3.411	-0.087	-0.463	-1.183	-1.902	-2.622	-3.341	-4.060
	3	0.322	0.241	0.000	-0.469	-0.938	-1.407	-1.875	0.352	0.249	-0.041	-0.597	-1.154	-1.710	-2.266
	4	0.796	0.941	0.864	0.695	0.407	0.119	-0.169	0.911	1.065	0.967	0.765	0.419	0.074	-0.272
	5	1.160	**1.324**	1.304	1.223	1.116	0.975	0.833	1.334	**1.528**	1.503	1.404	1.275	1.104	0.932
	6	1.090	1.147	1.128	1.085	1.033	0.975	0.910	1.266	1.331	1.304	1.251	1.185	1.113	1.032
	7	0.621	0.619	0.596	0.566	0.534	0.500	0.466	0.722	0.719	0.690	0.654	0.614	0.571	0.528

TABLE V
EMBEDDING RATES OF IMAGE $Crowd$ UNDER VARIOUS α AND β WHEN BLOCK SIZE IS 2×2 AND 3×3

$r_{\alpha,\beta}$ (bpp)		β (block size 2x2)							β (block size 3x3)						
		1	2	3	4	5	6	7	1	2	3	4	5	6	7
α	1	0.429	-0.174	-0.777	-1.379	-1.982	-2.585	-3.187	0.440	-0.276	-0.993	-1.709	-2.426	-3.143	-3.859
	2	0.693	0.564	0.072	-0.420	-0.912	-1.405	-1.897	0.724	0.581	-0.007	-0.596	-1.185	-1.774	-2.363
	3	1.107	1.162	0.984	0.638	0.292	-0.054	-0.400	1.207	1.272	1.059	0.640	0.221	-0.198	-0.617
	4	1.388	1.471	1.391	1.252	1.033	0.815	0.596	1.561	1.652	1.547	1.380	1.112	0.843	0.575
	5	1.414	**1.511**	1.485	1.414	1.324	1.208	1.093	1.616	**1.728**	1.691	1.603	1.492	1.348	1.203
	6	1.164	1.215	1.203	1.173	1.133	1.088	1.036	1.345	1.407	1.392	1.353	1.302	1.246	1.181
	7	0.649	0.654	0.638	0.617	0.594	0.570	0.545	0.755	0.761	0.742	0.716	0.688	0.659	0.628

TABLE VI
MAXIMUM EMBEDDING RATE COMPARISONS OF DIFFERENT IMAGES APPLIED BY PBTL–RDHEI AND SEVERAL RELATED ALGORITHMS

r_{max} (bpp)	Zhang [31]	Li [28]	Zhang [21]	Wu [32]	Yin [22]	Yin [23]	Zhou [9]	Huang [24]				Ma [11]	Cao [18]	PBTL-RDHEI	
								DHS_2	DHS_3	PHS_2	PHS_3			2×2	3×3
Lena	0.005	0.010	0.036	0.35	0.12	0.13	0.15	0.105	0.122	0.097	0.015	0.95	$\leqslant$0.8	1.722	2.018
Airplane	0.005	0.013	0.036	0.35	0.19	0.21	0.19	0.163	0.186	0.148	0.023	1.08	$\leqslant$1.4	1.903	2.188
Man	0.003	0.005	0.011	0.01	0.05	0.04	0.12	0.080	0.095	0.071	0.014	0.25	$\leqslant$1	1.327	1.528
Crowd	0.004	0.015	0.045	0.35	0.21	0.16	0.11	0.175	0.201	0.175	0.023	1.05	$\leqslant$1	1.511	1.728
kodim08	0.001	0.007	0.007	0.12	0.06	0.05	0.11	0.068	0.077	0.067	0.008	0.42	$\leqslant$0.5	0.783	0.849
kodim13	0.001	0.003	0.005	0.05	0.05	0.04	0.10	0.045	0.053	0.04	0.006	0.43	$\leqslant$0.6	0.698	0.801

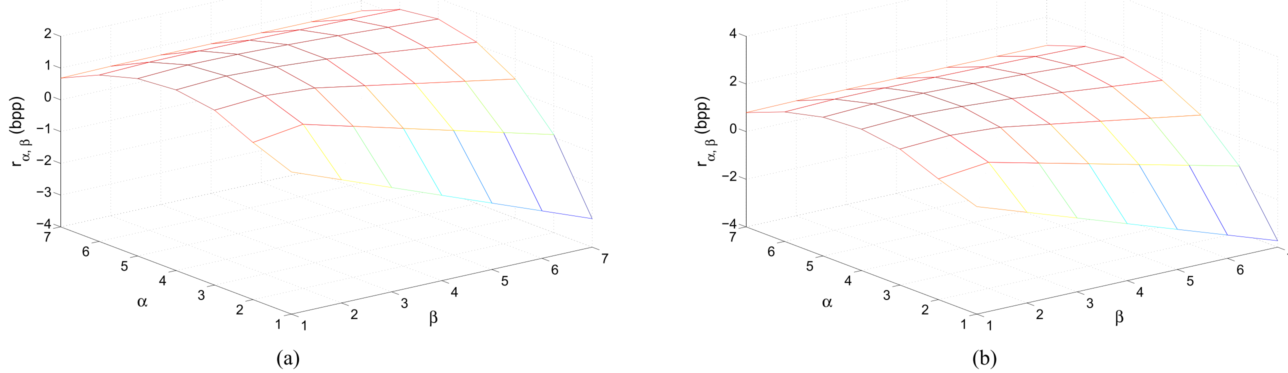


Fig. 17. Average embedding rates of 1000 marked encrypted images under various α and β when block sizes are (a) 2×2 and (b) 3×3.

TABLE VII
MAXIMUM EMBEDDING RATE OF TEST IMAGES WITH OR WITHOUT PIXEL MODULATION PROCESS UNDER DIFFERENT BLOCK SIZES

$r_{\max}$ (bpp)	2×2			3×3		
	with mod.	without mod.	reduced	with mod.	without mod.	reduced
Lena	1.722	1.746	0.054	2.018	2.084	0.066
Airplane	1.903	1.928	0.025	2.188	2.219	0.031
Man	1.327	1.372	0.045	1.528	1.583	0.055
Crowd	1.511	1.546	0.035	1.728	1.774	0.046
kodim08	0.783	0.822	0.039	0.849	0.907	0.058
kodim13	0.698	0.775	0.077	0.801	0.891	0.090
Aver1000	1.752	1.772	0.020	2.003	2.026	0.023

Table VI compares the maximum embedding rate of PBTL-RDHEI with several existing works. Because some related works may not fully reversible as analyzed in Section I, for fair of comparison, the result in Table VI is under the same situation that the secret data and original image can be fully extracted and recovered. In order to achieve a better performance, we set the block size to 4×4 in [22] and [23]. For Zhou *et al.* 's method [9], 3 bits of the secret data are embed into each encrypted image block for demonstration. In Huang *et al.*'s method [24], DHS_2(3) and PHS_2(3) indicate using difference histogram shifting and prediction-error expansion to embed secret data when block size is set to $2 \times 2 (3 \times 3)$, respectively. For PHS_2, the median edge detector is used and for PHS_3, the rhombus predictor is applied. For Ma *et al.*'s method [11], the 3 LSB planes are reserved for data embedding. As can be seen from the result, the proposed PBTL-RDHEI significantly improved the embedding rates.

To further analyze the embedding rate variation tendency under various α and β, we randomly select 1000 images from the BowsBase database to show the average embedding rates under different block sizes. The results are plotted in Fig. 17. From the results, we can observe that the embedding rates have similar variation tendency when block sizes are set to 2×2 and 3×3 under various α and β. When $\alpha = 4, \beta = 2$, images reach the maximum average embedding rates $r_{\max} = 1.722$ bpp and $r_{\max} = 2.018$ bpp when block sizes are set to 2×2 and 3×3, respectively.

The embedding ability of PBTL-DE depends on the spatial redundancy of pixels within blocks. In PBTL-RDHEI, the spatial redundancy of blocks in the original image is modified by pixel modulation using Eq. (9). In order to show the influence of pixel modulation process to the image spatial redundancy, we calculate the maximum embedding rate of some test images with and without pixel modulation. The results are listed in Table VII, where '$Aver1000$' indicates average result of 1000 images that randomly selected from the BowsBase database. From the results, we can observe that the modulation operation slightly reduced the embedding rates. Thus, the proposed image encryption well kept the spatial redundancy of the pixels within blocks.

VIII. Conclusion

In this paper, we first proposed a parametric binary tree labeling scheme (PBTL). Using PBTL, we then proposed a data embedding method (PBTL-DE) and further applied it into the encrypted images application (PBTL-RDHEI). The PBTL-DE is specific designed for encrypted-domain based application, because it significantly changed the pixel values in the image. PBTL-RDHEI is a full reversible method that both the secret data and original image can be extracted without any error. Experiment results and comparisons have shown that the PBTL-RDHEI significantly improved the embedding rate. Security analysis has demonstrated the robustness of PBTL-RDHEI in withstanding brute-force and know/chosen-plaintext attacks.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valued comments which helped to improve this paper.

References

[1] Y. Q. Shi, X. Li, X. Zhang, H. Wu, and B. Ma, "Reversible data hiding: Advances in the past two decades," *IEEE Access*, vol. 4, no. 99, pp. 3210–3237, May 2016.

[2] Z. Hua, Y. Zhou, C.-M. Pun, and C. L. P. Chen, "2D sine logistic modulation map for image encryption," *Inf. Sci.*, vol. 297, pp. 80–94, 2015.

[3] X. Zhang, "Reversible data hiding with optimal value transfer," *IEEE Trans. Multimedia*, vol. 15, no. 2, pp. 316–325, Feb. 2013.

[4] J. Fridrich, M. Goljan, and R. Du, "Lossless data embedding: New paradigm in digital watermarking," *EURASIP J. Appl. Signal Process.*, vol. 2002, no. 1, pp. 185–196, 2002.

[5] M. U. Celik, G. Sharma, A. M. Tekalp, and E. Saber, "Lossless generalized-LSB data embedding," *IEEE Trans. Image Process.*, vol. 14, no. 2, pp. 253–266, Feb. 2005.

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

[7] A. M. Alattar, "Reversible watermark using the difference expansion of a generalized integer transform," *IEEE Trans. Image Process.*, vol. 13, no. 8, pp. 1147–1156, Aug. 2004.

[8] Y. Hu, H.-K. Lee, and J. Li, "DE-based reversible data hiding with improved overflow location map," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 2, pp. 250–260, Feb. 2009.

[9] J. Zhou *et al.*, "Secure reversible image data hiding over encrypted domain via key modulation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 3, pp. 441–452, Mar. 2016.

[10] Z. Qian, X. Zhang, and S. Wang, "Reversible data hiding in encrypted JPEG bitstream," *IEEE Trans. Multimedia*, vol. 16, no. 5, pp. 1486–1491, Aug. 2014.

[11] K. Ma, W. Zhang, X. Zhao, N. Yu, and F. Li, "Reversible data hiding in encrypted images by reserving room before encryption," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 3, pp. 553–562, Mar. 2013.

[12] S. Yi and Y. Zhou, "Binary-block embedding for reversible data hiding in encrypted images," *Signal Process.*, vol. 133, pp. 40–51, 2017.

[13] X. Liao, K. Li, and J. Yin, "Separable data hiding in encrypted image based on compressive sensing and discrete fourier transform," *Multimedia Tools Appl.*, vol. 76, no. 20, pp. 20 739–20 753, 2017.

[14] X. Liao and C. Shu, "Reversible data hiding in encrypted images based on absolute mean difference of multiple neighboring pixels," *J. Vis. Commun. Image Represent.*, vol. 28, pp. 21–27, 2015.

[15] M. Li, D. Xiao, Y. Zhang, and H. Nan, "Reversible data hiding in encrypted images using cross division and additive homomorphism," *Signal Process.: Image Commun.*, vol. 39, pp. 234–248, 2015.

[16] X. Zhang, "Reversible data hiding in encrypted image," *IEEE Signal Process. Lett.*, vol. 18, no. 4, pp. 255–258, Apr. 2011.

[17] W. Zhang, K. Ma, and N. Yu, "Reversibility improved data hiding in encrypted images," *Signal Process.*, vol. 94, pp. 118–127, 2014.

[18] X. Cao, L. Du, X. Wei, D. Meng, and X. Guo, "High capacity reversible data hiding in encrypted images by patch-level sparse representation," *IEEE Trans. Cybern.*, vol. 46, no. 5, pp. 1132–1143, May 2016.

[19] S. Yi and Y. Zhou, "An improved reversible data hiding in encrypted images," in *Proc. IEEE China Summit Int. Conf. Signal Inf. Process.*, 2015, pp. 225–229.

[20] T. Mathew and M. Wilscy, "Reversible data hiding in encrypted images by active block exchange and room reservation," in *Proc. Int. Conf. Contemp. Comput. Inform.*, 2014, pp. 839–844.

[21] X. Zhang, "Separable reversible data hiding in encrypted image," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 2, pp. 826–832, Apr. 2012.

[22] Z. Yin, B. Luo, and W. Hong, "Separable and error-free reversible data hiding in encrypted image with high payload," *Scientific World J.*, vol. 2014, 2014, Art.ID. 604876.

[23] Z. Yin, H. Wang, H. Zhao, B. Luo, and X. Zhang, "Complete separable reversible data hiding in encrypted image," in *Proc. 1st Int. Conf. Cloud Comput. Security*, 2015, pp. 101–110.

[24] F. Huang, J. Huang, and Y. Q. Shi, "New framework for reversible data hiding in encrypted domain," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 12, pp. 2777–2789, Dec. 2016.

[25] Z. Qian and X. Zhang, "Reversible data hiding in encrypted images with distributed source encoding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 4, pp. 636–646, Apr. 2016.

[26] W. Puech, M. Chaumont, and O. Strauss, "A reversible data hiding method for encrypted images," in *Proc. SPIE Security, Forensics, Steganogr. Watermarking Multimedia Contents X*, 2008, pp. 68191E.

[27] W. Hong, T.-S. Chen, and H.-Y. 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.

[28] M. Li, D. Xiao, Z. Peng, and H. Nan, "A modified reversible data hiding in encrypted images using random diffusion and accurate prediction," *ETRI J.*, vol. 36, no. 2, pp. 325–328, 2014.

[29] X. Zhang, Z. Qian, G. Feng, and Y. Ren, "Efficient reversible data hiding in encrypted images," *J. Vis. Commun. Image Represent.*, vol. 25, no. 2, pp. 322–328, 2014.

[30] S. Zheng *et al.*, "Lossless data hiding algorithm for encrypted images with high capacity," *Multimedia Tools Appl.*, vol. 75, pp. 13765–13778, 2016.

[31] X. Zhang, C. Qin, and G. Sun, "Reversible data hiding in encrypted images using pseudorandom sequence modulation," *Digit. Forensics Watermaking*, vol. 7809, pp. 358–367, 2013.

[32] X. Wu and W. Sun, "High-capacity reversible data hiding in encrypted images by prediction error," *Signal Process.*, vol. 104, pp. 387–400, 2014.

[33] Y.-C. Chen, C.-W. Shiu, and G. Horng, "Encrypted signal-based reversible data hiding with public key cryptosystem," *J. Vis. Commun. Image Represent.*, vol. 25, no. 5, pp. 1164–1170, 2014.

[34] C.-W. Shiu, Y.-C. Chen, and W. Hong, "Encrypted image-based reversible data hiding with public key cryptography from difference expansion," *Signal Process.: Image Commun.*, vol. 39, pp. 226–233, 2015.

[35] X. Zhang, J. Wang, Z. Wang, and H. Cheng, "Lossless and reversible data hiding in encrypted images with public key cryptography," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 9, pp. 1622–1631, Sep. 2016.

[36] Z. Yin, A. Abel, J. Tang, X. Zhang, and B. Luo, "Reversible data hiding in encrypted images based on multi-level encryption and block histogram modification," *Multimedia Tools Appl.*, vol. 76, no. 3, pp. 3899–3920, 2017.

[37] Z. Hua, S. Yi, and Y. Zhou, "Medical image encryption using high-speed scrambling and pixel adaptive diffusion," *Signal Process.*, vol. 144, pp. 134–144, 2018.

[38] Y. Zhou, L. Bao, and C. P. Chen, "A new 1D chaotic system for image encryption," *Signal Process.*, vol. 97, pp. 172–182, 2014.

Shuang Yi received the B.E. degree in software engineering from Chongqing University, Chongqing, China. She is currently working toward the Ph.D. degree in the Department of Computer and Information Science, University of Macau, Macau, China. Her research interests include multimedia security and signal/image processing.

Yicong Zhou (M07–SM'14) received the B.S. degree from Hunan University, Changsha, China, and the M.S. and Ph.D. degrees from Tufts University, Massachusetts, U.S.A., all in electrical engineering. He is currently an Associate Professor and Director of the Vision and Image Processing Laboratory in the Department of Computer and Information Science, University of Macau, Macau, China. His research interests include chaotic systems, multimedia security, image processing and understanding, and machine learning.

He serves as an Associate Editor for *Neurocomputing*, *Journal of Visual Communication and Image Representation*, and *Signal Processing: Image Communication*. He is a co-chair of Technical Committee on Cognitive Computing in the IEEE Systems, Man, and Cybernetics Society. He was a recipient of the Third Price of Macau Natural Science Award, in 2014.