Separable Reversible Data Hiding for Encrypted Palette Images With Color Partitioning and Flipping Verification

Han-Zhou Wu, *Student Member, IEEE*, Yun-Qing Shi, *Fellow, IEEE*, Hong-Xia Wang, and Lin-Na Zhou

***Abstract*— Reversible data hiding (RDH) into encrypted images is of increasing attention to researchers as the original content can be perfectly reconstructed after the embedded data are extracted while the content owner's privacy remains protected. The existing RDH techniques are designed for grayscale images and, therefore, cannot be directly applied to palette images. Since the pixel values in a palette image are not the actual color values, but rather the color indexes, RDH in encrypted palette images is more challenging than that designed for normal image formats. To the best knowledge of the authors, there is no suitable RDH scheme designed for encrypted palette images that has been reported, while palette images have been widely utilized. This has motivated us to design a reliable RDH scheme for encrypted palette images. The proposed method adopts a color partitioning method to use the palette colors to construct a certain number of embeddable color triples, whose indexes are self-embedded into the encrypted image so that a data hider can collect the usable color triples to embed the secret data. For a receiver, the embedded color triples can be determined by verifying a self-embedded check code that enables the receiver to retrieve the embedded data only with the data hiding key. Using the encryption key, the receiver can roughly reconstruct the image content. Experiments have shown that our proposed method has the property that the presented data extraction and image recovery are separable and reversible. Compared with the state-of-the-art works, our proposed method can provide a relatively high data-embedding payload, maintain high peak signal-to-noise ratio values of the decrypted and marked images, and have a low computational complexity.**

***Index Terms*— Encryption, palette, partitioning, reversible data hiding (RDH), separable, watermarking.**

I. Introduction

REVERSIBLE data hiding (RDH) [1] aims to embed secret bits into an innocent object (e.g., digital image) by slightly altering the insignificant components of a cover signal, and the hidden data as well as the original content should be recovered from the marked content without any loss. Since RDH allows the original content to be completely recovered, as a means of information hiding, the RDH techniques are quite desirable and helpful in some sensitive applications such as remote sensing, multimedia archive management, and military communication. There are some important requirements for RDH methods [1]–[5]: the data embedding capacity may be high, the quality of marked image should be high, the computational complexity should be low, and the security level should be high. For a fixed payload, we expect to reduce the distortion as much as possible to obtain a better image quality, while for a required distortion level we hope to maximize the payload. In the meantime, an RDH algorithm with a lower complexity and a higher security level corresponds to a more efficient usage in applications. It is noted that the RDH is a fragile technique [5], meaning that when the marked image is manipulated, e.g., lossy compressed, one will find that it is not authentic, and the image content cannot be fully recovered.

Many RDH methods [1]–[23] have been reported in past years. Early RDH methods [2]–[4] utilize lossless compression to create extra space for embedding. Later on, more efficient techniques have been introduced to increase the embedding capacity and keep the distortion low, such as difference expansion (DE) [5], histogram shifting (HS) [1], and prediction-error expansion [6]–[9]. As the DE can provide a high payload, various methods have been developed along this line to improve the embedding performance, such as the pixel prediction mechanism [6]–[9], [12], [13], generalized integer transform [10], [11], and improvement of compressing a location map [14]. In the HS-based methods, the message bits are embedded into the host image by shifting the histogram bins. As the HS can possibly significantly reduce the embedding distortion and provide a sufficient payload, many HS-based methods [1], [7], [15]–[19] have been reported to ensure a good payload-distortion behavior. Furthermore, RDH [20]–[23] in terms of the payload limit subjected to a given distortion has been also designed to approach the theoretical bound.

In some applications, e.g., cloud service, the content owner may hope to share an image only with an authorized recipient. Thus, the content owner may encrypt the original image before transmission [24]–[27]. Meanwhile, the network manager/server expects to embed extra data into the encrypted image though he cannot access the image content. It can be

Manuscript received June 26, 2015; revised November 11, 2015, January 14, 2016, and March 3, 2016; accepted March 28, 2016. Date of publication April 20, 2016; date of current version August 2, 2017. This work was supported by the National Natural Science Foundation of China under Grant U1536110 and Grant U1536207. This paper was recommended by Associate Editor W. Zeng.

H.-Z. Wu and H.-X. Wang are with the School of Information Science and Technology, Southwest Jiaotong University, Chengdu 611756, China (e-mail: wuhanzhou_2007@126.com; hxwang@home.swjtu.edu.cn).

Y.-Q. Shi is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102-1982 USA (e-mail: shi@njit.edu).

L.-N. Zhou is with the University of International Relations, Beijing 100091, China (e-mail: zhoulinna@tsinghua.edu.cn).

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

Digital Object Identifier 10.1109/TCSVT.2016.2556585

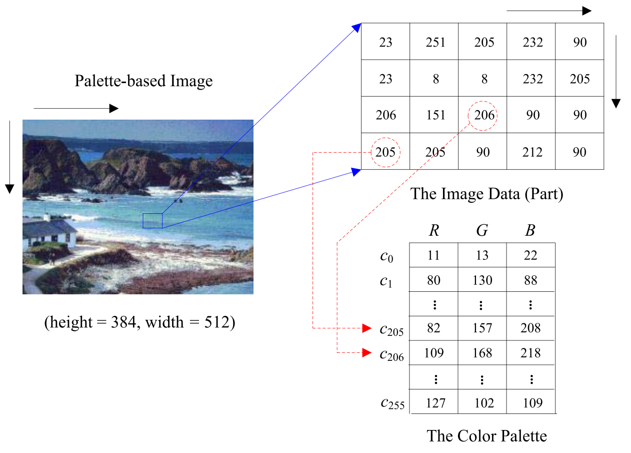


Fig. 1. Example to illustrate a palette-based image. Here, a pixel value of 205 points to the color at c_{205}, indicating that the actual color value of the pixel is (82, 157, 208).

seen that RDH in encrypted images is an intuitive and effective way to meet such a requirement. The above RDH techniques work for unencrypted content and cannot be applied to encrypted domain directly. In [24], by flipping three LSBs of a half of pixels in an encrypted block, 1 b can be embedded. Both data extraction and image recovery proceed with the aid of spatial correlations of pixels, which may produce an error rate. To reduce the error rate, Hong *et al.* [25] take into account the block smoothness as well as pixel correlations in the border of neighboring blocks. In [26], the message bits are embedded into a sparse space reserved by compressing the LSBs of pixels in an encrypted image. In order to improve the payload, Zhang *et al.* [27] use a half of the fourth LSB plane in an encrypted image as the space to hide data. As JPEG is widely used, Qian *et al.* [28] further introduce an RDH scheme in the encrypted JPEG images. Since the entropy of an encrypted image is maximized, it is difficult to losslessly create room for data hiding. To overcome this drawback, Ma *et al.* [29] present a scheme that uses a traditional RDH method to reserve room before image encryption, which maintains that both data extraction and image recovery are lossless. Thereafter, an RDH method [30] using a prediction technique is introduced. In the method, the secret data are embedded by altering the prediction errors in an encrypted image. It can be said that a predictor that well exploits the pixel correlations will benefit the method. To better explore the pixel correlations, Cao *et al.* [31] propose to hide the additional data with a patch-level sparse representation technique. According to the sparse coding technique, a larger vacated room can be constructed, and thus it outperforms some state-of-the-art works in terms of payload-distortion behavior.

It is true that the above methods have moved the research on RDH ahead rapidly. It should be noted that, however, most of them work for grayscale images and a very few are suitable for palette images (e.g., GIF and PNG-8), which use a color palette to provide efficient storage for images with a limited number of colors, e.g., charts, computer art, and color quantized true color images [32]. Fig. 1 shows an example of a palette-based image. Different from grayscale images, for a palette image, there are a color list called *color palette* and a pixel matrix (or index matrix) called *image data*. The color palette consists of a list of selected colors with red, green, and blue (RGB) components. And the image data specifies the color indexes of pixels. Since palette images are very popular in modern multimedia systems and Internet applications [32]–[34], RDH techniques applied to palette images are desirable. However, as the pixel values are not the actual color values, the RDH for palette images is more challenging than that for images in other image formats. Though some RDH schemes [34], [35] have been designed for palette images, they cannot be directly applied to the encrypted domain.

Inspired by the perspective to separable RDH (SRDH) [26], in this paper, we present a reliable SRDH scheme for encrypted palette images. The proposed method aims to divide the palette colors into multiple color triples, among which the embeddable color triples are recorded and self-embedded into the encrypted index matrix together with some other auxiliary data before the image transmission so that secret bits can be carried by altering the pixel values of embeddable color triples on the data hider side. After a receiver acquires the encrypted and marked image, he should extract the self-embedded auxiliary data at first. The subsequent procedure depends on the role of the receiver. If he has only the data hiding key, he can retrieve the hidden data without knowing the image content. He can decrypt the marked and encrypted image with a good image quality if he has only the encryption key. If the receiver has both the data hiding key and encryption key, he can not only retrieve the hidden data but also recover the image without loss. In summary, the proposed method ensures that the data extraction and image recovery are separable and reversible. Compared with the related works, our method does not produce an error rate in the data extraction and image recovery. Experiments have also shown that our method can provide a relatively high embedding payload and maintain a very good quality of the decrypted and marked image.

The remainder of this paper is organized as follows. Section II shows the general framework of the proposed method. Then, Section III gives the detailed RDH procedure. Experiments and analysis are presented in Section IV. Finally, the conclusion is given in Section V.

II. General Framework

From the data embedding point of view, in general, the RDH in encrypted images can be classified into two categories: 1) directly embed secret bits into an encrypted image [24]–[28] and 2) first reserve space for appending extra data before image encryption, and then embed secret bits into the vacated space in the encrypted image [29]–[31]. In applications such as cloud service, instead of the content owner, the content manager may serve as a data hider. The content owner expects to send only an encrypted image to the manager without extra information; the manager needs to embed extra data into the encrypted image. It can be seen that Category 1 corresponds to a more intuitive use of the RDH for encrypted images. However, as the entropy of an encrypted image is maximized, it is difficult to losslessly vacate room for data embedding [31].

Therefore, Category 2 corresponds to a more effective way to provide a high payload.

On the other hand, from the data extraction point of view, the RDH in encrypted images can be classified into two categories: 1) directly retrieve the secret data in the marked and encrypted image with only the data hiding key and 2) decrypt the marked and encrypted image as well as extract the secret data with both the data hiding key and encryption key. Considering the above scenario, the content owner and the content manager may not play the role of each other. It means that the content manager can extract the secret data, while the image content is unknown to him, and the content owner can get an approximation image of the original one, while has no access to the secret data. Thus, Category 1 seems to be more intuitive and desirable.

Therefore, an RDH method in encrypted images corresponds to a more efficient use if it possesses three characteristics.

1) The content owner knows the encryption key, and the data hider has the data hiding key.
2) The content owner or an authorized receiver can roughly recover the image content, while knows nothing about the secret data; the data extractor can retrieve the secret data, while cannot access image content.
3) One can fully recover the secret data and image content when he has both the decryption key and data hiding key. Our method is designed for encrypted palette images, which meets the above requirements.

The proposed method mainly consists of four phases, i.e., the encrypted image generation, data embedding, data extraction, as well as image recovery. In the encrypted image generation, by employing the color partitioning algorithm, a certain number of usable color triples are collected. Then, both the color palette and index matrix are encrypted, respectively. The indexes for the collected color triples are thereafter self-embedded into the encrypted image together with some other auxiliary data. In this way, an encrypted image with self-embedded data is generated. After obtaining the encrypted image, a data hider can embed the secret data into the encrypted image. The idea of data hiding process is to modify the pixel values of collected color triples. After a receiver acquires the marked and encrypted image, he should extract the auxiliary data at first. Then, he can fully retrieve the secret data if he has only the data hiding key. With only the encryption key, he can produce an approximation image of the original one. The receiver can fully reconstruct the image content and retrieve the secret data if he has both the encryption key and data hiding key. In summary, the data extraction and image recovery are separable and reversible. Fig. 2 shows the framework of the proposed method. We will describe the details in Section III.

III. Proposed Method

We will describe our RDH algorithm with the following three aspects: 1) the encrypted image generation; 2) data hiding in the encrypted images; and 3) data extraction and image recovery.

A. Encrypted Image Generation

To generate an embeddable encrypted image, three phases: 1) color partitioning; 2) image encryption; and 3) self-embedding, are involved. Let $\mathbf{I}\ \{\mathbf{P};\ \mathbf{C}\}$ be the palette image, where $\mathbf{P}$ is the index matrix sized $N \times M$ and $\mathbf{C}$ is the color palette sized $n \times 3$. Here, n denotes the number of palette colors. At first, we adopt a color partitioning method to divide colors in $\mathbf{C}$ into multiple color triples, which are then classified into *nonembeddable* or *embeddable*. According to a secret key, we encrypt $\mathbf{P}$ and $\mathbf{C}$ as $\mathbf{P}_E$ and $\mathbf{C}_E$, respectively. Then, the indexes of the embeddable color triples are self-embedded into $\mathbf{I}\{\mathbf{P}_E;\ \mathbf{C}_E\}$ together with a flipping check code and a location map. In this way, a required encrypted image $\mathbf{I}\{\mathbf{P}_{ES};\ \mathbf{C}_E\}$ can be obtained.

1) Color Partitioning: In order to collect color triples from $\mathbf{C}$, the occurrence frequency of each palette color is determined by

$$h(\mathbf{C}[k]) = \sum_{i=0}^{N-1}\sum_{j=0}^{M-1} \rho(k, \mathbf{P}[i, j]), \quad (0 \le k < n) \qquad (1)$$

where $\mathbf{C}[k]$ denotes the kth color in $\mathbf{C}$, $\mathbf{P}[i, j]$ is the pixel value at pixel position (i, j), and $\rho(k, \mathbf{P}[i, j]) = 1$ for the case $\mathbf{P}[i, j] = k$; otherwise $\rho(k, \mathbf{P}[i, j]) = 0$.

To evaluate the difference between any two RGB colors $c_1(r_1, g_1, b_1)$ and $c_2(r_2, g_2, b_2)$, we here define the Euclidean distance between c_1 and c_2 as the measurement, namely

$$\|c_1 - c_2\| = \sqrt{(r_1 - r_2)^2 + (g_1 - g_2)^2 + (b_1 - b_2)^2}. \qquad (2)$$

In general, a lower $||c_1 - c_2||$ indicates a lower visual difference between c_1 and c_2. The color partitioning procedure aims to partition colors in $\mathbf{C}$ into $\lfloor n/3 \rfloor$ color triples, where $\lfloor * \rfloor$ denotes the floor function. An iterative method is employed to construct the $\lfloor n/3 \rfloor$ color triples. In each iteration, three palette colors are orderly selected to constitute a color triple. Clearly, let S denote the resultant color triple set and be empty initially. A color with the highest occurrence frequency is first selected by

$$c_h = \arg\max_{e \in \mathbf{C}} h(e). \qquad (3)$$

A color with the lowest occurrence frequency is then selected by

$$c_a = \arg\min_{e \in \mathbf{C} \text{ and } e \neq c_h} h(e). \qquad (4)$$

Third, a color with the lowest difference to c_a is selected by

$$c_b = \arg\min_{e \in \mathbf{C}, e \neq c_h \text{ and } e \neq c_a} \|e - c_a\|. \qquad (5)$$

We then remove c_h, c_a, and c_b from $\mathbf{C}$ and append (c_h, c_a, c_b) to S, denoted by

$$S = S \cup \{(c_h, c_a, c_b)\}. \qquad (6)$$

By repeating the above-mentioned procedure, we can finally construct S. It is noted that during the process to obtain S, the original index position of every selected color can be recorded so that the original $\mathbf{C}$ can be reconstructed. It indicates

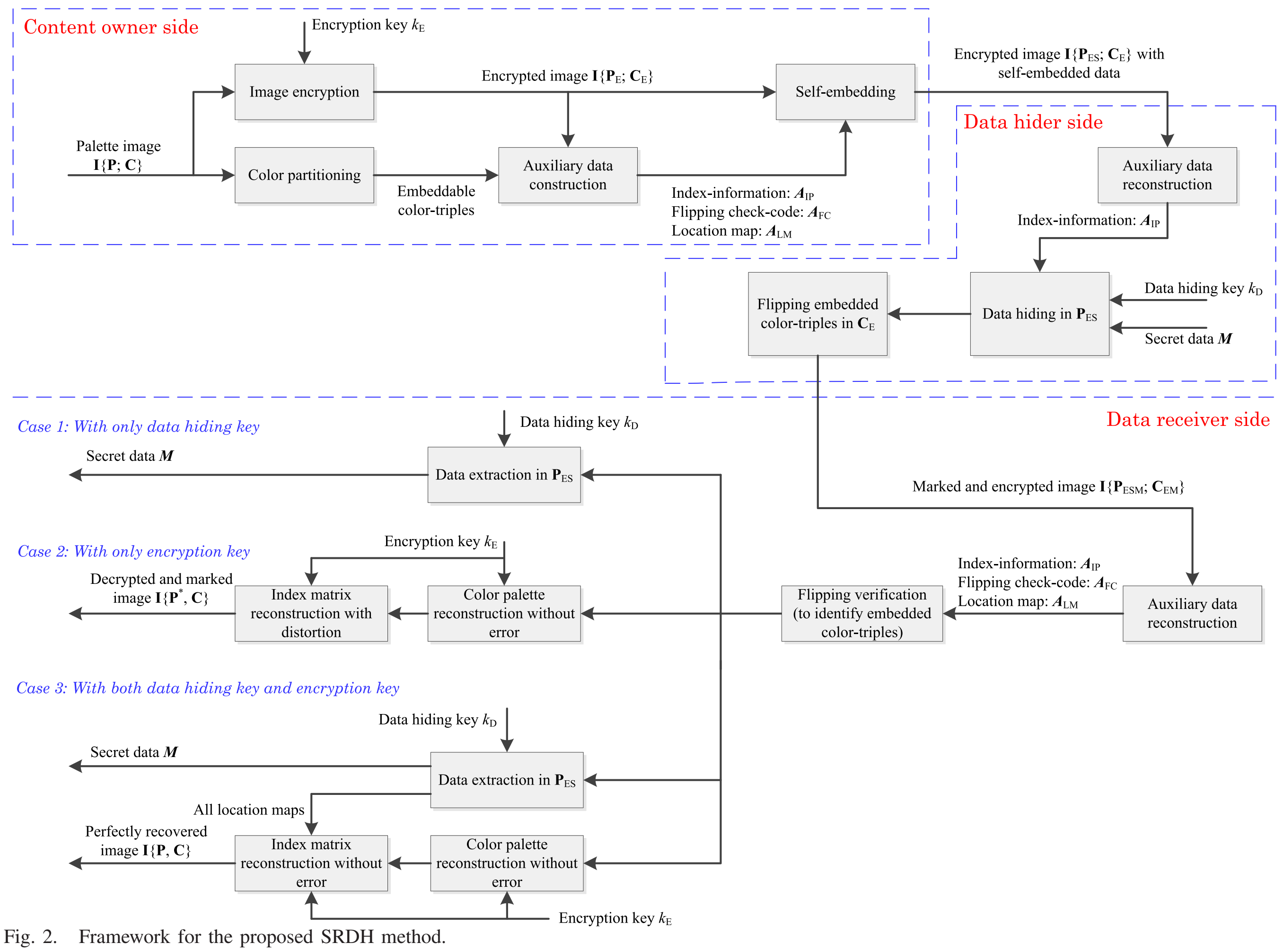

Content owner side
Encryption key k_E
Image encryption
Encrypted image $I\{P_E; C_E\}$
Self-embedding
Palette image $I\{P; C\}$
Color partitioning
Embeddable color-triples
Auxiliary data construction
Index-information: A_{IP}
Flipping check-code: A_{FC}
Location map: A_{LM}
Encrypted image $I\{P_{ES}; C_E\}$ with self-embedded data
Data hider side
Auxiliary data reconstruction
Index-information: A_{IP}
Data hiding in P_{ES}
Data hiding key k_D
Secret data M
Flipping embedded color-triples in C_E
Data receiver side
Case 1: With only data hiding key
Data hiding key k_D
Secret data M
Data extraction in P_{ES}
Marked and encrypted image $I\{P_{ESM}; C_{EM}\}$
Case 2: With only encryption key
Encryption key k_E
Decrypted and marked image $I\{P^*, C\}$
Index matrix reconstruction with distortion
Color palette reconstruction without error
Flipping verification (to identify embedded color-triples)
Index-information: A_{IP}
Flipping check-code: A_{FC}
Location map: A_{LM}
Auxiliary data reconstruction
Case 3: With both data hiding key and encryption key
Data hiding key k_D
Secret data M
Data extraction in P_{ES}
All location maps
Perfectly recovered image $I\{P, C\}$
Index matrix reconstruction without error
Color palette reconstruction without error
Encryption key k_E

Fig. 2. Framework for the proposed SRDH method.

that the color partitioning method essentially does not alter $\mathbf{I}\{\mathbf{P}; \mathbf{C}\}$. To enable a data hider to hide secret data in the encrypted domain, we classify the color triples as *embeddable* or *nonembeddable*. A color triple (c_h, c_a, c_b) is classified as embeddable if $h(c_h) > \lceil \log_2 NM \rceil \times (1 + h(c_a))$, where $\lceil * \rceil$ indicates the ceiling function; otherwise, it is nonembeddable. Accordingly, the elements in S can be classified as either embeddable or nonembeddable. We collect all the embeddable color triples from S to form a subset S_E. We then sort the elements in S_E according to $h(c_h)$ and $h(c_a)$. Without the loss of generality, let L and (c_h^k, c_a^k, c_b^k) be the number of color triples in S_E and the kth $(0 \leq k < L)$ element in S_E, respectively. Meanwhile, $h(c_h^0) \geq h(c_h^1) \geq \cdots \geq h(c_h^{L-1})$ and $h(c_a^0) \leq h(c_a^1) \leq \cdots \leq h(c_a^{L-1})$. For c_h^k, c_a^k, and c_b^k $(0 \leq k < L)$, let d_h^k, d_a^k, and d_b^k be their index positions in $\mathbf{C}$. For example, if c_h^k, c_a^k, and c_b^k are corresponding to $\mathbf{C}[117]$, $\mathbf{C}[138]$, and $\mathbf{C}[69]$, respectively, we will have $d_h^k = 117$, $d_a^k = 138$, and $d_b^k = 69$. Therefore, (c_h^k, c_a^k, c_b^k) is equivalent to $(\mathbf{C}[d_h^k], \mathbf{C}[d_a^k], \mathbf{C}[d_b^k])$ for all $0 \leq k < L$.

2) Image Encryption: After obtained S_E, $\mathbf{C}$ is encrypted with a cryptographic algorithm, denoted by (7), where k_{EC} is the secret key for encrypting $\mathbf{C}$. Clearly, we compute the bit stream of each $\mathbf{C}[k]$ $(0 \leq k < n)$ at first. For example, if $\mathbf{C}[k]$ is a 24-b RGB color (12, 7, 32), the bitstream will be $(00001100, 0000\ 0111, 00100000)_2$. Then, we concatenate the bit-streams of all $\mathbf{C}[k]$ $(0 \leq k < n)$ to constitute a larger bitstream, which is then encrypted with a cryptographic algorithm. Finally, we use the encrypted bitstream to replace the bitstream of every $\mathbf{C}[k]$ $(0 \leq k < n)$ by an inverse operation. In [24]–[31], the XOR operation is used to encrypt the cover image with an encryption key. Here, the encryption algorithm does not rely on the XOR operation. It means that we can use other encryption methods, e.g., the block cipher, which is more applicable to practical applications. We encrypt $\mathbf{P}$ by permuting all elements, denoted by (8), where k_{EP} is the secret key, to improve the security. Cleary, we could collect all $\mathbf{P}[i, j]$ $(0 \leq i < N, 0 \leq j < M)$ to constitute a sequence and permute the sequence using k_{EP}. We then use the permuted sequence to form $\mathbf{P}_E$ with an inverse operation. Although the permutation-based encryption for $\mathbf{P}$ has a lower security level than, e.g., the block cipher, it is almost impossible to figure out the permutation to recover $\mathbf{P}$ since there are a number of ways to permute $\mathbf{P}$, especially for $\mathbf{P}$ with a very large size. Therefore, we can encrypt $\mathbf{I}\{\mathbf{P}; \mathbf{C}\}$ as $\mathbf{I}\{\mathbf{P}_E; \mathbf{C}_E\}$ with an encryption key k_E, which consists of k_{EC} and k_{EP}

$$\mathbf{C}_E = \text{Enc}\,(\mathbf{C}, k_{\text{EC}}) \tag{7}$$

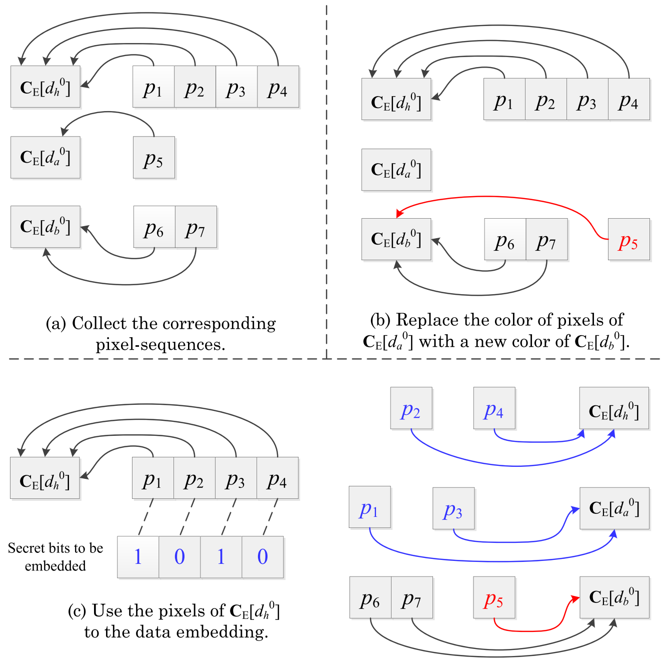
$$\mathbf{P}_E = \text{Perm}\,(\mathbf{P}, k_{\text{EP}})\,. \tag{8}$$

3) Self-Embedding: For data hiding in the encrypted domain, we need to reversibly self-embed some auxiliary data into $\mathbf{I}\{\mathbf{P}_E; \mathbf{C}_E\}$. It enables a data hider to use S_E to hide secret data. Though $(\mathbf{C}[d_h^k], \mathbf{C}[d_a^k], \mathbf{C}[d_b^k])$ is encrypted as $(\mathbf{C}_E[d_h^k], \mathbf{C}_E[d_a^k], \mathbf{C}_E[d_b^k])$, they both correspond to (c_h^k, c_a^k, c_b^k) as they have the identical index positions. The index positions (d_h^k, d_a^k, d_b^k), $(0 \le k < L)$, will be self-embedded into $\mathbf{P}_E$. Let $\boldsymbol{A}_{\text{IP}}$ be the index information to be embedded. We employ (c_h^0, c_a^0, c_b^0), i.e., $(\mathbf{C}_E[d_h^0], \mathbf{C}_E[d_a^0], \mathbf{C}_E[d_b^0])$, to embed $\boldsymbol{A}_{\text{IP}}$ as it can provide the largest payload. It is noted that for natural images, it is enough to carry $\boldsymbol{A}_{\text{IP}}$ using (c_h^0, c_a^0, c_b^0) according to our experiments. It is straightforward to apply more color triples if (c_h^0, c_a^0, c_b^0) cannot carry $\boldsymbol{A}_{\text{IP}}$.

In order to ensure data extraction and image recovery, some additional auxiliary data will be embedded as well. Clearly, the self-embedded information contains three parts: 1) $\boldsymbol{A}_{\text{IP}}$; 2) a flipping check code $\boldsymbol{A}_{\text{FC}}$; and 3) a location map $\boldsymbol{A}_{\text{LM}}$. Specifically, $\lceil \log_2 \frac{n}{3} \rceil + 3L \cdot \lceil \log_2 n \rceil$ bits are required to store $\boldsymbol{A}_{\text{IP}}$, where $\lceil \log_2 \frac{n}{3} \rceil$ bits are stipulated to store L, and $3L \cdot \lceil \log_2 n \rceil$ bits are used to store (d_h^k, d_a^k, d_b^k) for $0 \le k < L$. For each $(\mathbf{C}_E[d_h^k], \mathbf{C}_E[d_a^k], \mathbf{C}_E[d_b^k])$, $(0 \le k < L)$, we compute the parity bits of RGB components of each color. For an integer number, the parity bit is defined as 0 if it is even; otherwise, its parity bit is 1. We apply the XOR operation to the parity bits of a color triple. For example, for a RGB color triple $\{(80, 131, 96), (20, 11, 43), (21, 14, 46)\}$, the parity bits are $\{(0, 1, 0), (0, 1, 1), (1, 0, 0)\}$ and the XOR value can be computed as 0. In this way, we can collect the XOR values of $(\mathbf{C}_E[d_h^k], \mathbf{C}_E[d_a^k], \mathbf{C}_E[d_b^k])$ for all $0 \le k < L$. We orderly concatenate the values to form $\boldsymbol{A}_{\text{FC}}$. In obvious, the size of $\boldsymbol{A}_{\text{FC}}$ is equal to L. It can be inferred that if we flip the bitstream of a color triple, the XOR value will be changed. For example, the flipped version of three 24-b RGB colors $\{(80, 131, 96), (20, 11, 43), (21, 14, 46)\}$ is $\{(175, 124, 159), (235, 244, 212), (234, 241, 209)\}$, which corresponds to a new XOR value of 1.

$\boldsymbol{A}_{\text{LM}}$ records all the pixels in $\mathbf{P}_E$ that have a value of d_a^0. Thus, we need $\lceil \log_2 NM \rceil \times (1 + h(\mathbf{C}_E[d_a^0]))$ bits to store $\boldsymbol{A}_{\text{LM}}$, where $\lceil \log_2 NM \rceil$ bits are used to tell $h(\mathbf{C}_E[d_a^0])$ and the rest are used to record the $h(\mathbf{C}_E[d_a^0])$ pixels. $\boldsymbol{A}_{\text{IP}}$, $\boldsymbol{A}_{\text{FC}}$, and $\boldsymbol{A}_{\text{LM}}$ are embedded by modifying the pixels of $\mathbf{C}_E[d_h^0]$ and $\mathbf{C}_E[d_a^0]$. Clearly, all the pixels in $\mathbf{P}_E$ that have a value of d_a^0 are first replaced with a value of d_b^0. Then, some pixels of $\mathbf{C}_E[d_h^0]$ are modified to carry $(\boldsymbol{A}_{\text{IP}}, \boldsymbol{A}_{\text{FC}}, \boldsymbol{A}_{\text{LM}})$ and each will carry one bit. Specifically, for a pixel with a value of d_h^0, if the secret bit is 0, the pixel will be unchanged; otherwise, the pixel value will be modified as d_a^0. After the self-embedding, we can obtain the self-embedded $\mathbf{P}_E$, denoted by $\mathbf{P}_{\text{ES}}$. Thus, the required encrypted image $\mathbf{I}\{\mathbf{P}_{\text{ES}}; \mathbf{C}_E\}$ can be finally generated.

To explain the self-embedding procedure, we take Fig. 3 for example. As shown in Fig. 3(a), we collect the pixel sequences of $(\mathbf{C}_E[d_h^0], \mathbf{C}_E[d_a^0], \mathbf{C}_E[d_b^0])$, e.g., the pixel p_5 specifies a color of $\mathbf{C}_E[d_a^0]$ and p_3 specifies a color of $\mathbf{C}_E[d_h^0]$. We then replace all the pixels of $\mathbf{C}_E[d_a^0]$ with a color of $\mathbf{C}_E[d_b^0]$, which is shown in Fig. 3(b). In Fig. 3(c), we use the pixels of $\mathbf{C}_E[d_h^0]$ to carry 1010. Specifically, p_1 is modified with a color of $\mathbf{C}_E[d_a^0]$ to carry 1 and p_2 is kept unchanged so as to carry 0. Both p_3 and p_4 are processed with the similar way. Therefore, we can embed 1010 into the pixels.



$\mathbf{C}_E[d_h^0]$
p_1 p_2 p_3 p_4
$\mathbf{C}_E[d_a^0]$
p_5
$\mathbf{C}_E[d_b^0]$
p_6 p_7
(a) Collect the corresponding pixel-sequences.
(b) Replace the color of pixels of $\mathbf{C}_E[d_a^0]$ with a new color of $\mathbf{C}_E[d_b^0]$.
Secret bits to be embedded
1 0 1 0
(c) Use the pixels of $\mathbf{C}_E[d_h^0]$ to the data embedding.

Fig. 3. Example to explain the self-embedding using a color triple. (a) Collecting the corresponding pixel sequences. (b) Replacing the color of pixels of $\mathbf{C}_E[d_a^0]$ with a new color of $\mathbf{C}_E[d_b^0]$. (c) Using the pixels of $\mathbf{C}_E[d_h^0]$ to the data embedding.

B. Data Hiding in Encrypted Images

After acquiring $\mathbf{I}\{\mathbf{P}_{\text{ES}}; \mathbf{C}_E\}$, a data hider should first extract $\boldsymbol{A}_{\text{IP}}$, $\boldsymbol{A}_{\text{FC}}$, and $\boldsymbol{A}_{\text{LM}}$ from $\mathbf{P}_{\text{ES}}$ so as to embed extra information. It is noted that like the methods in [29]–[31], it essentially requires a user–server/user–manager interaction [36], i.e., a data hider has to identify the vacated embedding room with the auxiliary data provided by the content owner. All the usable color triples can be identified with $\boldsymbol{A}_{\text{IP}}$. The secret data $\boldsymbol{M}$ will be embedded by modifying the pixels of these usable color triples, which is similar to the self-embedding process. Suppose that we have extracted $\boldsymbol{A}_{\text{IP}}$, $\boldsymbol{A}_{\text{FC}}$, and $\boldsymbol{A}_{\text{LM}}$ from $\mathbf{I}\{\mathbf{P}_{\text{ES}}; \mathbf{C}_E\}$. It implies that we can compute (d_h^k, d_a^k, d_b^k) for $0 \le k < L$. In the following, we orderly process each color triple until $\boldsymbol{M}$ is fully carried.

For the 0th color triple, though it has been used to carry $\boldsymbol{A}_{\text{IP}}$, $\boldsymbol{A}_{\text{FC}}$, and $\boldsymbol{A}_{\text{LM}}$, there may be extra pixels of $\mathbf{C}_E[d_h^0]$ that have not been embedded. In this case, a data hider can apply these pixels for data hiding. Since the location map has been self-embedded, we employ the unembedded pixels to carry only $\boldsymbol{M}_0$, which is a part of $\boldsymbol{M}$. It is noted that if $(\mathbf{C}_E[d_h^0], \mathbf{C}_E[d_a^0], \mathbf{C}_E[d_b^0])$ cannot carry extra data, it means that the size of $\boldsymbol{M}_0$ is 0. For the kth $(1 \le k < L)$ color triple, the idea of data embedding is to modify the pixels of $\mathbf{C}_E[d_h^k]$ and $\mathbf{C}_E[d_a^k]$. The information to be embedded consists of two parts: a location map $\boldsymbol{O}_k$ and $\boldsymbol{M}_k$ (a part of $\boldsymbol{M}$).

Similarly, $\boldsymbol{O}_k$ records all the pixels in $\mathbf{P}_{\text{ES}}$ that have a value of d_a^k. We can employ $\lceil \log_2 NM \rceil$ bits to store $h(\mathbf{C}_E[d_a^k])$, and $\lceil \log_2 NM \rceil \times h(\mathbf{C}_E[d_a^k])$ bits to record the $h(\mathbf{C}_E[d_a^k])$ pixels. $\lceil \log_2 NM \rceil \times (1 + h(\mathbf{C}_E[d_a^k]))$ bits are therefore required. The bits are carried by modifying such pixels that have a value of d_h^k. As a pixel carries one bit, the size of $\boldsymbol{M}_k$ will be at most $h(\mathbf{C}_E[d_h^k]) - \lceil \log_2 NM \rceil \times (1 + h(\mathbf{C}_E[d_a^k]))$. It implies that $(\mathbf{C}_E[d_h^k], \mathbf{C}_E[d_a^k], \mathbf{C}_E[d_b^k])$ can carry extra data only if $h(\mathbf{C}_E[d_h^k]) > \lceil \log_2 NM \rceil \times (1 + h(\mathbf{C}_E[d_a^k]))$. That is the reason we compute S_E from S during the encrypted image generation.

After $\boldsymbol{O}_k$ and $\boldsymbol{M}_k$ are orderly embedded according to the data hiding key, the bit stream of $(\mathbf{C}_E[d_h^k], \mathbf{C}_E[d_a^k], \mathbf{C}_E[d_b^k])$ will be flipped, e.g., $(10010100, 01001011, 11110100)_2$ is the flipped version of the RGB color $(01101011, 10110100, 00001011)_2$. It ensures that an authorized receiver can identify the embedded color triples. By repeating the embedding operations for more color triples, $\boldsymbol{M}$ can be embedded into $\mathbf{I}\{\mathbf{P}_E; \mathbf{C}_E\}$ to produce the marked and encrypted image $\mathbf{I}\{\mathbf{P}_{\text{ESM}}; \mathbf{C}_{\text{EM}}\}$, where both $\mathbf{P}_{\text{ESM}}$ and $\mathbf{C}_{\text{EM}}$ are an altered version of $\mathbf{P}_{\text{ES}}$ and $\mathbf{C}_E$, respectively.

Based on the above analysis, we are now ready to describe the data hiding procedure in encrypted images as follows.

1) *Step 1:* Extract $(\boldsymbol{A}_{\text{IP}}, \boldsymbol{A}_{\text{FC}}, \boldsymbol{A}_{\text{LM}})$ from $\mathbf{P}_{\text{ES}}$, and determine (d_h^k, d_a^k, d_b^k), $0 \le k < L$, according to $\boldsymbol{A}_{\text{IP}}$. Let L_E be the actual number of embedded color triples. For $(\mathbf{C}_E[d_h^k], \mathbf{C}_E[d_a^k], \mathbf{C}_E[d_b^k])$, $0 \le k < L_E$, perform Steps 2–4 so that $\boldsymbol{M}$ and $\boldsymbol{O}_k$ $(1 \le k \le L_E)$ could be embedded into $\mathbf{I}\{\mathbf{P}_{\text{ES}}; \mathbf{C}_E\}$. Note that to protect the secret data securely, $\boldsymbol{M}$ is always been encrypted. Moreover, all $\boldsymbol{M}_k$ $(0 \le k < L_E)$ can be concatenated to constitute $\boldsymbol{M}$.
2) *Step 2:* If the color triple is $(\mathbf{C}_E[d_h^0], \mathbf{C}_E[d_a^0], \mathbf{C}_E[d_b^0])$, we collect the unembedded pixels of $\mathbf{C}_E[d_h^0]$, and embed $\boldsymbol{M}_0$ into the pixels according to the data hiding key k_D. Otherwise, we first construct $\boldsymbol{O}_k$ and modify all pixel values of $\mathbf{C}_E[d_a^k]$ as d_b^k and then embed $\boldsymbol{O}_k$ and $\boldsymbol{M}_k$ into the pixels of $\mathbf{C}_E[d_h^k]$ with k_D. During the data embedding, for a pixel to be embedded, if the secret bit equals 0, it will be unchanged; otherwise, the pixel value will be replaced with d_a^k.
3) *Step 3:* Flip $(\mathbf{C}_E[d_h^k], \mathbf{C}_E[d_a^k], \mathbf{C}_E[d_b^k])$ and remove $\boldsymbol{M}_k$ from $\boldsymbol{M}$. If $\boldsymbol{M}$ is empty, perform Step 4; otherwise, perform Step 2.
4) *Step 4:* With an inverse operation, use the modified pixels to form the marked $\mathbf{P}_{\text{ES}}$. With the marked $\mathbf{P}_{\text{ES}}$ and the flipped $\mathbf{C}_E$, construct the marked and encrypted image $\mathbf{I}\{\mathbf{P}_{\text{ESM}}; \mathbf{C}_{\text{EM}}\}$.

On the receiver side, the data extraction and image recovery are separable and reversible. The steps to extract $\boldsymbol{M}$ and recover an approximation image will be introduced in the following.

C. Data Extraction and Image Recovery

In the following, we will consider the cases that a receiver has only k_D, only k_E, and both k_D and k_E, respectively.

1) Data Extraction Only With Data Hiding Key: With $\mathbf{I}(\mathbf{P}_{\text{ESM}}; \mathbf{C}_{\text{EM}})$ and k_D, a receiver can extract $\boldsymbol{M}$. It can be performed with an inverse operation of the data embedding process. At first, we extract $(\boldsymbol{A}_{\text{IP}}, \boldsymbol{A}_{\text{FC}}, \boldsymbol{A}_{\text{LM}})$ from $\mathbf{P}_{\text{ESM}}$. Then, according to $\boldsymbol{A}_{\text{IP}}$, we reconstruct (d_h^k, d_a^k, d_b^k) for all $0 \le k < L$. Since the bitstream of a color triple will be flipped if it is used for data hiding, the data extractor can identify the embedded color triples by comparing $\boldsymbol{A}_{\text{FC}}$ with the new XOR values of $(\mathbf{C}_{\text{EM}}[d_h^k], \mathbf{C}_{\text{EM}}[d_a^k], \mathbf{C}_{\text{EM}}[d_b^k])$ for all $0 \le k < L$. The reason is that if a color triple was flipped, the new XOR value will be different from the original one. Therefore, the data extractor can determine L_E and further extract $\boldsymbol{M}_k$ from the marked pixels of $\mathbf{C}_{\text{EM}}[d_h^k]$ for each $0 \le k < L_E$. Thus, $\boldsymbol{M}$ can be reconstructed and further decrypted as the original message. The detailed steps are described as follows.

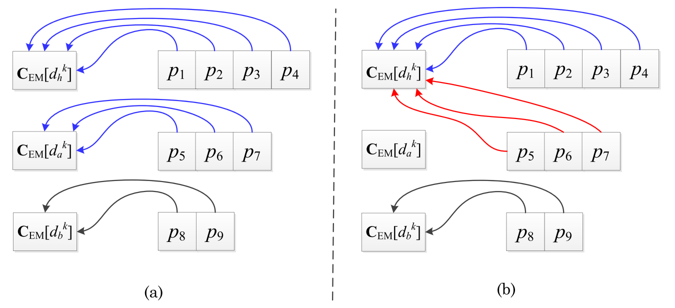

$\mathbf{C}_{\text{EM}}[d_h^k]$
p_1
p_2
p_3
p_4
$\mathbf{C}_{\text{EM}}[d_a^k]$
p_5
p_6
p_7
$\mathbf{C}_{\text{EM}}[d_b^k]$
p_8
p_9
(a)
(b)

Fig. 4. Example to show the recovery process for an embedded color triple. (a) Collecting the corresponding pixel sequences on the receiver side. (b) Replacing the color of pixels of $\mathbf{C}_{\text{EM}}[d_a^k]$ with its original color of $\mathbf{C}_{\text{EM}}[d_h^k]$.

1) *Step 1*: Extract $(\boldsymbol{A}_{\text{IP}}, \boldsymbol{A}_{\text{FC}}, \boldsymbol{A}_{\text{LM}})$ and compute (d_h^k, d_a^k, d_b^k), $0 \le k < L$, according to $\boldsymbol{A}_{\text{IP}}$.
2) *Step 2*: For each $(\mathbf{C}_{\text{EM}}[d_h^k], \mathbf{C}_{\text{EM}}[d_a^k], \mathbf{C}_{\text{EM}}[d_b^k])$, $0 \le k < L$, compute the XOR value. By comparing all the new XOR values with $\boldsymbol{A}_{\text{FC}}$, identify the embedded color triples and L_E.
3) *Step 3:* For each $0 \le k < L_E$, collect all the pixels of $\mathbf{C}_{\text{EM}}[d_h^k]$ and $\mathbf{C}_{\text{EM}}[d_a^k]$ from $\mathbf{P}_{\text{ESM}}$. Extract 0 for a pixel corresponding to $\mathbf{C}_{\text{EM}}[d_h^k]$ and 1 for that corresponding to $\mathbf{C}_{\text{EM}}[d_a^k]$. With k_D, use the extracted bits to retrieve $\boldsymbol{M}_k$ $(0 \le k < L_E)$ without error.
4) *Step 4:* Concatenate $\boldsymbol{M}_k$ for all $0 \le k < L_E$, to form $\boldsymbol{M}$ and finally decrypt $\boldsymbol{M}$ as the original secret message.

2) Image Recovery Only With Encryption Key: In this case, though the receiver cannot recover the original secret message, he can roughly recover the image content. After extracting $(\boldsymbol{A}_{\text{IP}}, \boldsymbol{A}_{\text{FC}}, \boldsymbol{A}_{\text{LM}})$, L_E can be computed. Then, by reflipping $(\mathbf{C}_{\text{EM}}[d_h^k], \mathbf{C}_{\text{EM}}[d_a^k], \mathbf{C}_{\text{EM}}[d_b^k])$, $0 \le k < L_E$, $\mathbf{C}_E$ can be reconstructed from $\mathbf{C}_{\text{EM}}$. With k_{EC}, $\mathbf{C}$ can be further decrypted from $\mathbf{C}_E$.

On the other hand, for $(\mathbf{C}_{\text{EM}}[d_h^k], \mathbf{C}_{\text{EM}}[d_a^k], \mathbf{C}_{\text{EM}}[d_b^k])$, $0 \le k < L_E$, the pixels with a value of d_a^k in $\mathbf{P}_{\text{ESM}}$ should be modified as their original value d_h^k. Fig. 4 shows an example to process the pixels of an embedded color triple. As shown in Fig. 4(a), the marked pixels are $\{p_1, p_2, p_3, p_4, p_5, p_6, p_7\}$. It can be inferred that the pixels $\{p_5, p_6, p_7\}$ correspond to d_h^k in $\mathbf{I}\{\mathbf{P}; \mathbf{C}\}$. Therefore, to generate an approximation image, as shown in Fig. 4(b), $\{p_5, p_6, p_7\}$ are modified with a value of d_h^k. For $(\mathbf{C}_{\text{EM}}[d_h^0], \mathbf{C}_{\text{EM}}[d_a^0], \mathbf{C}_{\text{EM}}[d_b^0])$, according to $\boldsymbol{A}_{\text{LM}}$, the pixels that have an original value of d_a^0 can be identified from the pixel set of $\mathbf{C}_{\text{EM}}[d_b^0]$ in $\mathbf{P}_{\text{ESM}}$. These pixels can be modified with a pixel value of d_a^0. In this way, an index matrix $\mathbf{P}_E^*$, can be reconstructed. According to k_{EP}, $\mathbf{P}_E^*$ can be decrypted as $\mathbf{P}^*$, which is approximated to $\mathbf{P}$.

Therefore, a palette image $\mathbf{I}\{\mathbf{P}^*; \mathbf{C}\}$ can be generated. It can be seen that the image recovery procedure results in a distorted index matrix $\mathbf{P}^*$, while can recover the color palette without loss. The distortion between $\mathbf{P}^*$ and $\mathbf{P}$ depends on such pixels that have an original value of d_a^k in $\mathbf{P}$, while have a new value of d_b^k in $\mathbf{P}^*$ for $1 \le k < L_E$. Since $||\mathbf{C}[d_a^k] - \mathbf{C}[d_b^k]||$ is minimized, the visual difference between $\mathbf{I}\{\mathbf{P}; \mathbf{C}\}$ and $\mathbf{I}\{\mathbf{P}^*; \mathbf{C}\}$ will be small, implying that a good image quality can be achieved.

3) Data Extraction and Image Recovery With Both Keys: In the case that a receiver has both k_E and k_D, he can retrieve the original secret message and reconstruct $\mathbf{I}\{\mathbf{P}; \mathbf{C}\}$ without error. At first, the receiver retrieves the original secret message from $\mathbf{I}\{\mathbf{P}_{\text{ESM}}; \mathbf{C}_{\text{EM}}\}$ by applying the above data extraction procedure. Obviously, $\boldsymbol{O}_k$ $(1 \le k < L_E)$ can be reconstructed as well.

According to the above image recovery process, $\mathbf{C}$ can be recovered without error and $\mathbf{P}_E^*$ can be computed as well. With $\boldsymbol{O}_k$ for all $1 \le k < L_E$, the receiver can reconstruct $\mathbf{P}_E$ from $\mathbf{P}_E^*$. By decrypting $\mathbf{P}_E$ as $\mathbf{P}$, $\mathbf{I}\{\mathbf{P}; \mathbf{C}\}$ can be finally obtained.

In the following section, we will present our experiments and analysis to evaluate the performance of the proposed method.

IV. Experiments and Analysis

In this section, we first analyze our method in terms of the confidentiality and reliability. Then, experiments are presented to evaluate the payload-distortion behavior. The computational complexity in practical applications is finally discussed.

A. Confidentiality and Reliability

From the content owner's point of view, it is demanded that the image content should not be obtained by any unauthorized receiver. As $\mathbf{C}$ was encrypted as $\mathbf{C}_E$ before transmission, none can obtain the actual color of each pixel without the encryption key. In [24]–[31], the XOR operation is used to encrypt the cover image. For our method, it does not rely on the XOR operation, indicating that other more secure encryption algorithms can be used for encrypting $\mathbf{C}$. Meanwhile, the index matrix $\mathbf{P}$ was also pseudorandomly permuted so as to improve the confidentiality. Although the permutation-based encryption provides a lower security level than the block cipher, it is almost impossible to figure out the permutation to recover the image content since there are a large number of ways of permutation, especially for images with a large size. Therefore, the encryption process can provide satisfactory confidentiality as an unauthorized decoder will hardly access the image visual content.

To enable the data hider to embed extra data in the encrypted domain, some auxiliary data should be self-embedded into $\mathbf{P}_E$. Like the methods in [29]–[31], it actually requires a user–manager interaction [36], i.e., the data hider has to identify the vacated room with the auxiliary data provided by the content owner. To extract the auxiliary data, one should identify (d_h^0, d_a^0, d_b^0) and the self-embedding key at first. As it will require extra storage space, a content owner

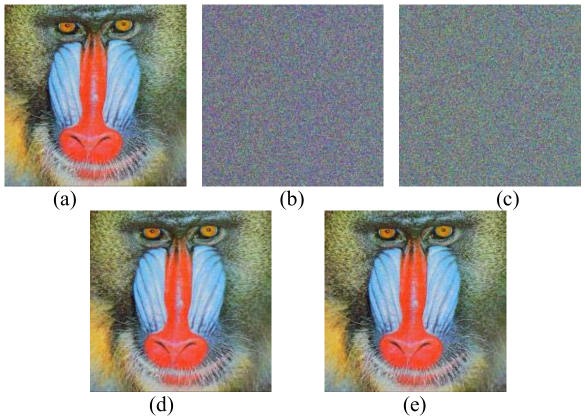


Fig. 5. Experiments for the *Baboon* image (GIF format). (a) Original image. (b) Encrypted image containing self-embedded data. (c) Encrypted and marked image. (d) Decrypted and marked image with a PSNR value of 50.13 dB. (e) Recovered image without any loss.

may embed the parameters into the *reserved field* (namely, *alpha-channel* [37], [38]) of $\mathbf{I}\{\mathbf{P}_E; \mathbf{C}_E\}$, which causes no destruction. Also, it is straightforward for the content owner to empty some specified pixels and use these pixels to store the parameters as long as these pixels are kept unchanged throughout subsequent procedure, and the original pixel values should be self-embedded as well. Since the size of the self-embedding parameters is very small, the impact on the data embedding payload can be ignored. It can be inferred that the process will still maintain separability and reversibility.

For the data hider, after a color triple has been embedded, the bitstream of the color triple should be flipped, which results in that the required XOR value of the color triple will be changed. Since $\boldsymbol{A}_{\text{FC}}$ shows the original values, all embedded color triples can be identified by verifying the new XOR values with $\boldsymbol{A}_{\text{FC}}$ on the receiver side. Accordingly, a data extractor can fully extract the hidden information with the aid of k_D. However, in [24] and [25], the hidden data are extracted by taking account into the local texture, which may produce an error rate in data extraction. In the proposed method, the data extraction process is independent of the image texture such that the data extraction operations are free of any error, which maintains the integrity of hidden data.

B. Embedding Payload and Distortion

We use the peak signal-to-noise ratio (PSNR) to evaluate the distorted image quality. The formula of the PSNR is defined as

$$\text{PSNR} = 10 \cdot \log_{10} \frac{255^2}{\text{MSE}} (dB) \tag{9}$$

$$\text{MSE} = \frac{1}{3} \cdot \frac{1}{N \times M} \cdot \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \|I_{i,j} - J_{i,j}\|^2 \tag{10}$$

where $I_{i,j}$ and $J_{i,j}$ denote the colors of the original image and the recovered image at pixel position (i, j), respectively. A larger PSNR value usually indicates a better image quality.

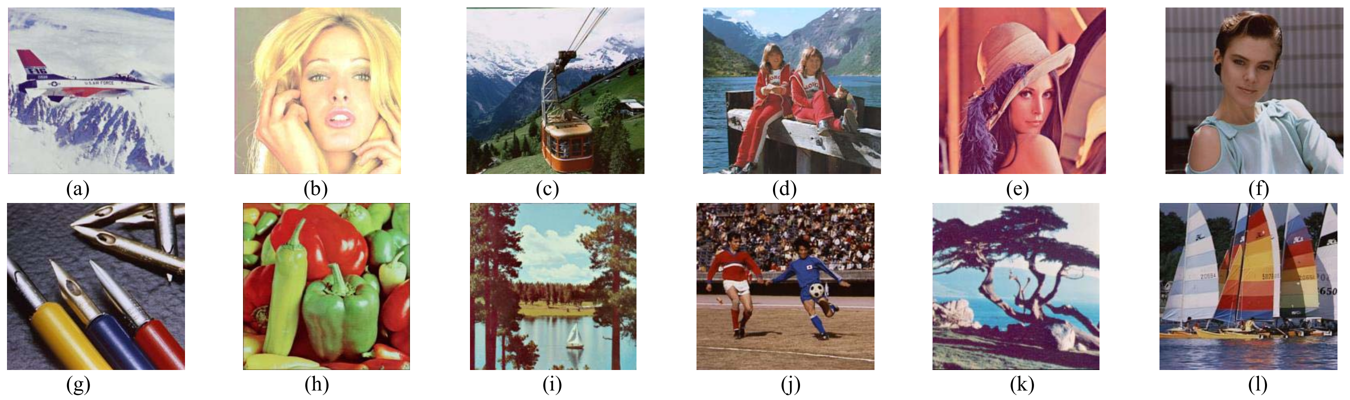


Fig. 6. Twelve testing images (all with 256 colors). (a) *Airplane*, 512 × 512. (b) *Tiffany*, 512 × 512. (c) *Cable-car*, 512 × 480. (d) *Kids*, 512 × 480. (e) *Lena*, 512 × 512. (f) *Model*, 512 × 480. (g) *Pens*, 512 × 480. (h) *Peppers*, 512 × 512. (i) *Sailboat*, 512 × 512. (j) *Soccer*, 512 × 480. (k) *Trees*, 256 × 256. (l) *Yacht*, 512 × 480.

TABLE I
DATA EMBEDDING CAPACITY FOR 12 TESTING IMAGES

Image	Entire Capacity (bits)	Pure Capacity (bits)	L_E
Airplane	25111	14545	8
Tiffany	62085	26193	15
Cable-car	72452	57278	11
Kids	22763	13135	7
Lena	10592	5336	4
Model	92258	71229	18
Pens	53673	28294	16
Peppers	132235	89665	32
Sailboat	6223	3973	2
Soccer	31490	19974	8
Trees	13872	9680	14
Yacht	22066	11028	6

TABLE II
PSNR (db) FOR THE DECRYPTED AND MARKED IMAGES WITH THE MAXIMUM EMBEDDING PAYLOAD

Image	PSNR	Image	PSNR
Airplane	57.60	Pens	55.74
Tiffany	51.72	Peppers	48.77
Cable-car	57.97	Sailboat	61.67
Kids	58.48	Soccer	59.48
Lena	63.78	Trees	58.36
Model	56.58	Yacht	59.95

We first take the *Baboon* image of size 512 × 512 (a total of 256 colors) to demonstrate the feasibility of our method. A total of 167 239 message bits, i.e., an entire embedding rate of 0.64 bpp, can be successfully embedded into the encrypted image, where $L_E = L = 47$. As the location maps $\boldsymbol{O}_k$ ($1 \leq k < L_E$) are utilized to reconstruct the image content, the pure embedding rate is 0.57 bpp (i.e., a total of 150 229 b). Fig. 5 shows the experiments for *Baboon*. It can be seen that the encrypted image containing self-embedded data maintains the invisibility of the image content. Fig. 5(d) shows the decrypted and marked image with a high PSNR value of 50.13 dB. It can be seen that our method does not introduce notable artifacts. With the aid of the location maps, as shown in Fig. 5(e), the original image can be reconstructed without distortion.

Twelve testing images shown in Fig. 6 are further utilized to show the performance of the proposed method. Table I shows the data embedding capacity for each testing image. As shown in Table I, the *Peppers* image achieves a high pure embedding capacity of 89 665 message bits, while the *Sailboat* image has a relatively low pure embedding capacity of 3973 message bits. It is seen that, the application of the proposed method to different images will result in different data embedding capacity. And an image with a larger number of usable color triples usually provides a relatively larger capacity. In fact, the capacity also depends on the occurrence frequencies of the used colors in a color triple, i.e., for $(\mathbf{C}_E[d_h^k], \mathbf{C}_E[d_a^k], \mathbf{C}_E[d_b^k])$, $1 \leq k < L_E$, a larger difference between $h(\mathbf{C}_E[d_h^k])$ and the size of $\boldsymbol{O}_k$ will facilitate the data embedding capacity. Though different images provide a different embedding capacity, all the decrypted and marked images can maintain a high level of the image quality, as shown in Table II. To demonstrate the payload-distortion performance, the testing images have been further applied to data hiding with different embedding rates, which has been shown in Fig. 7. It can be observed that the proposed method maintains a low introduced distortion for the decrypted and marked images.

Because the SRDH for encrypted palette images is very rare, we here compare our method with some of the recently reported methods. Fig. 8 shows the payload-distortion performance of different approaches on the *Lena*, *Man*, *Airplane*, and *Crowd* images (all sized 512 × 512). It can be seen that the methods introduced in [29] and [31] can provide a relatively higher capacity than the methods introduced in [24]–[28] and [30] and our method. Though the proposed method has a lower embedding capacity than the methods introduced in [29] and [31], the proposed

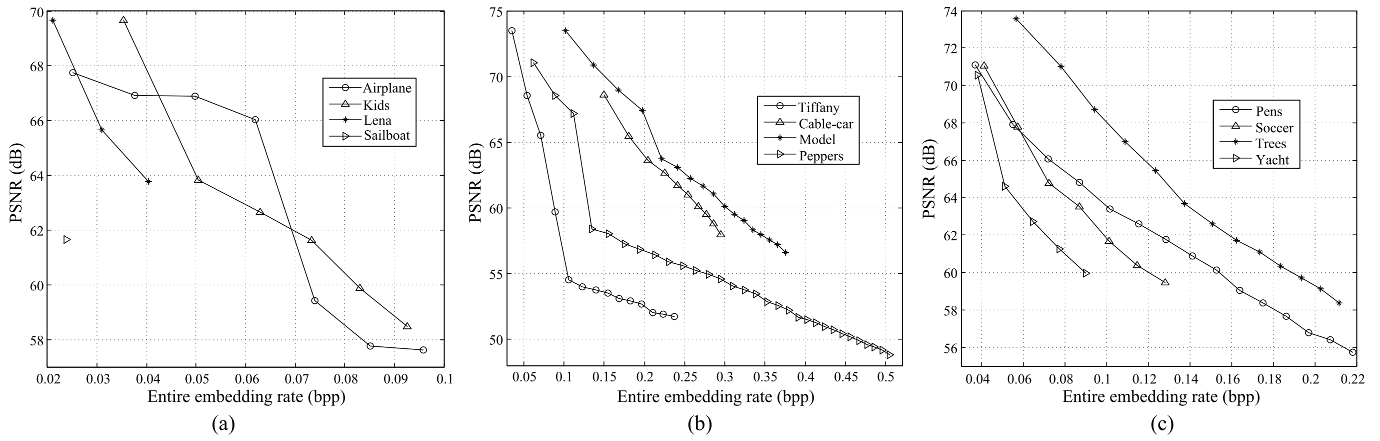


Fig. 7. Payload-distortion performance of the 12 testing images. (a) *Airplane*, *Kids*, *Lena*, and *Sailboat*. (b) *Tiffany*, *Cable-car*, *Model*, and *Peppers*. (c) *Pens*, *Soccer*, *Trees*, and *Yacht*.

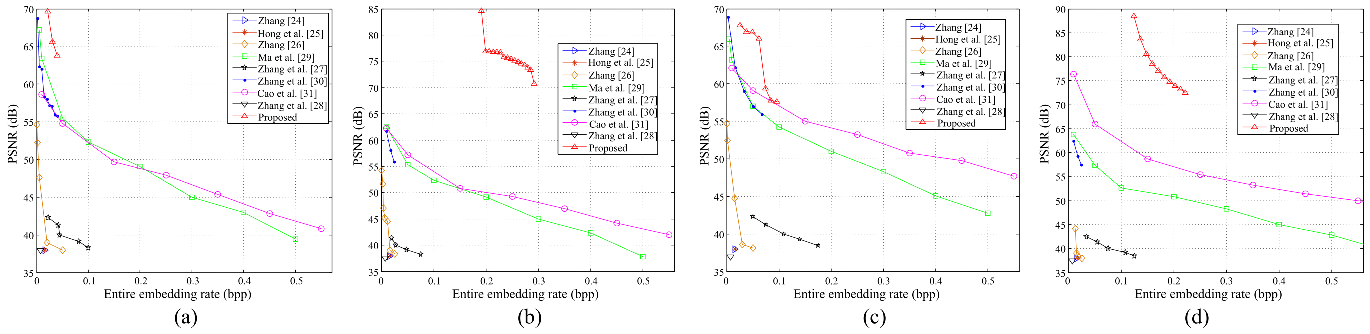


Fig. 8. Comparison of the payload-distortion performance between different methods on four testing images. (a) *Lena*. (b) *Man*. (c) *Airplane*. (d) *Crowd*.

method can maintain high PSNR values. It indicates that the proposed method can provide a relatively low distortion to the decrypted and marked image. Compared with the methods in [24] and [25], since our method does not rely on pixel correlations in both data extraction and image recovery, our method does not produce an error rate in the data extraction and image recovery.

In applications, a palette image may have a palette with a few colors, e.g., 128-color. In order to evaluate the performance in images with a small size of color palette, we take the *Baboon* and *Peppers* with a different number of colors for experiments. The testing images with a few colors were transformed from the original 256-color images using the CxImage Library with the default settings (http://www.xdp.it/cximage.htm). Table III shows the embedding capacity and the PSNR with the maximum embedding payload. Fig. 9 gives the decrypted and marked image with the maximum payload, which indicates that our method does not introduce obvious artifacts. It can be seen that an image with a larger color palette often results in a larger number of embeddable color triples. And a larger color palette usually maintains a relatively larger embedding capacity and a relatively lower image distortion. For a larger color palette, as it has more palette colors, the size of S will be larger, which has the potential for providing a larger L_E. In fact, in addition to the size of a color palette, L_E also relies

TABLE III

EMBEDDING CAPACITY (b) AND PSNR (db) FOR THE *Baboon* AND *Peppers* WITH A DIFFERENT NUMBER OF PALETTE COLORS

Image (512×512)	Entire capacity	Pure capacity	L_E	PSNR
Baboon (32 colors)	49975	36997	2	45.28
Baboon (64 colors)	85590	66852	5	48.04
Baboon (128 colors)	115220	84602	15	45.68
Baboon (256 colors)	167239	150229	47	50.13
Peppers (32 colors)	67773	35949	3	40.66
Peppers (64 colors)	117731	87545	7	46.90
Peppers (128 colors)	143800	110014	18	46.92
Peppers (256 colors)	132235	89665	32	48.77

on the values of $h(\mathbf{C}_E[d_h^k])$ and $h(\mathbf{C}_E[d_a^k])$. Fig. 10 shows the values of $h(\mathbf{C}_E[d_h^k])$ and $h(\mathbf{C}_E[d_a^k])$ for $0 \leq k < L_E$. It could be observed that a smaller color palette usually corresponds to a larger $h(\mathbf{C}_E[d_h^k])$ and a larger $h(\mathbf{C}_E[d_a^k])$. In general, a larger $h(\mathbf{C}_E[d_h^k])$ allows more bits to be embedded. However, a larger $h(\mathbf{C}_E[d_a^k])$ requires more additional bits to store the location maps, which is not beneficial to the pure embedding payload. In addition, for

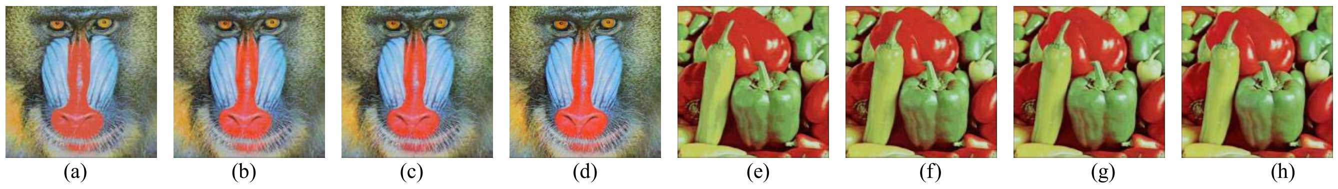


Fig. 9. Resultant decrypted and marked image with the maximum embedding payload (all with a size of 512 × 512). (a) *Baboon*—32 colors and PSNR: 45.28 dB. (b) *Baboon*—64 colors and PSNR: 48.04 dB. (c) *Baboon*—128 colors and PSNR: 45.68 dB. (d) *Baboon*—256 colors and PSNR: 50.13 dB. (e) *Peppers*—32 colors and PSNR: 40.66 dB. (f) *Peppers*—64 colors and PSNR: 46.90 dB. (g) Peppers—128 colors and PSNR: 46.92 dB. (h) Peppers—256 colors and PSNR: 48.77 dB.

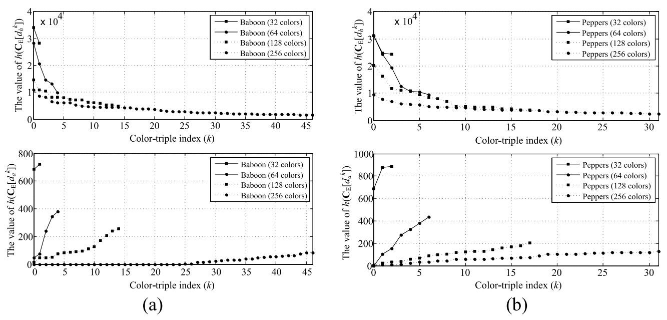


Fig. 10. Values of $h(\mathbf{C}_E[d_h^k])$ and $h(\mathbf{C}_E[d_a^k])$ for $0 \le k < L_E$. (a) *Baboon*. (b) *Peppers*.

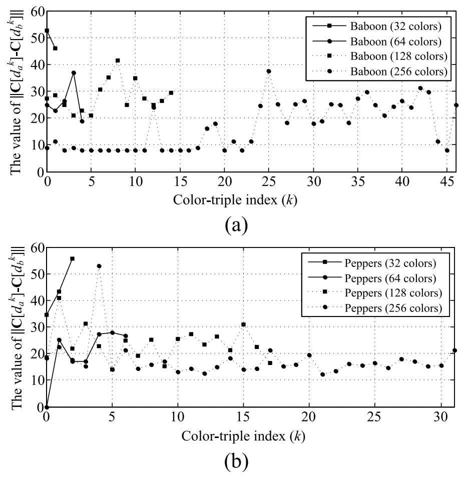


Fig. 11. Distortion between $\mathbf{C}[d_a^k]$ and $\mathbf{C}[d_b^k]$ for $0 \le k < L_E$. (a) *Baboon*. (b) *Peppers*.

an image, when k increases, $h(\mathbf{C}_E[d_h^k])$ will decline and $h(\mathbf{C}_E[d_a^k])$ will increase. Therefore, a smaller palette is expected to provide a relatively smaller L_E as well as a relatively smaller capacity. In general, to reduce the image distortion, $||\mathbf{C}[d_a^k] - \mathbf{C}[d_b^k]||$ is required to be as low as possible. Fig. 11 shows $||\mathbf{C}[d_a^k] - \mathbf{C}[d_b^k]||$ for $0 \le k < L_E$. It can be observed that, on the whole, a smaller color palette usually results in a higher distortion between $\mathbf{C}[d_a^k]$ and $\mathbf{C}[d_b^k]$, which implies that for a required payload, a smaller color palette is likely to introduce a relatively higher distortion. Overall, as shown in Fig. 12, our algorithm can maintain a good payload-distortion performance for images with a different size of color palette and the PSNRs are above 40 dB.

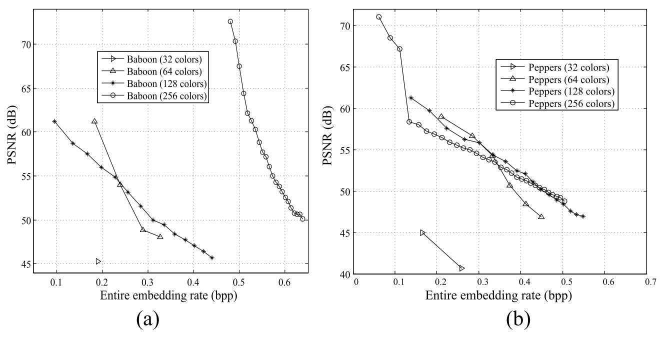


Fig. 12. Payload-distortion performance for the (a) *Baboon* and (b) *Peppers* images with a different number of palette colors.

C. Computational Complexity Analysis

For the computational complexity, here we mainly consider the three steps: 1) color partitioning; 2) image encryption; and 3) data embedding with the embeddable color triples. For color partitioning, the computational complexity could be denoted by $O(N \cdot M) + O(n^2) + O(L \cdot \log_2 L)$. The first item is the complexity to compute the occurrence frequencies. The second item denotes the complexity to collect the embeddable color triples, and the last item represents the complexity to sort the color triples. For the image encryption, the content owner has to encrypt $\mathbf{C}$ and permute $\mathbf{P}$. A secure cryptographic approach such as AES can be applied to quickly encrypt $\mathbf{C}$. Meanwhile, we employ the permutation technique reported in [39] to permute $\mathbf{P}$ since it requires very low extra information and maintains a near-linear complexity of the image size. In order to embed secret data into $\mathbf{P}_{\mathrm{ES}}$, the pixels of all the color triples to be embedded should be collected in advance. It is implemented within a complexity of $O(N \cdot M)$. In this way, one can visit the pixels to be embedded as well as modify the pixel values within a linear complexity of the number of the pixels. It implies that the secret data can be embedded within a very low computational complexity. On the receiver side, the subsequent operations can be considered as an inverse process, which maintains a low computational cost.

V. Discussion and Conclusion

RDH in encrypted images is a promising research topic that has drawn increasing attention because of its practical usages. In recent years, some state-of-the-art works have been reported. However, most work with grayscale images and cannot be directly applied to palette images. Palette images are known to have a relatively small size, which can reduce storage space and transmission time. This paper introduces a reliable SRDH method for encrypted palette images. The data hider can benefit from the data-embedding space reserved by the color partitioning procedure and apply the color replacement method to embed the additional data. Experiments have shown that the data extraction and image recovery are separable and are free of any error. And the proposed method can provide a satisfactory embedding payload, as well as maintain high PSNR values for the directly decrypted image, which indicates that the proposed method has a good potential for practical applications.

Acknowledgment

The authors would like to thank the the anonymous reviewers for their worthy comments and suggestions.

References

[1] 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.

[2] J. Fridrich, M. Goljan, and R. Du, "Invertible authentication," *Proc. SPIE*, vol. 4314, pp. 197–208, Aug. 2001.

[3] J. Fridrich, M. Goljan, and R. Du, "Lossless data embedding: New paradigm in digital watermarking," *EURASIP J. Appl. Signal Process.*, vol. 2, pp. 185–196, Feb. 2002.

[4] M. U. Celik, G. Sharma, and A. M. Tekalp, "Lossless watermarking for image authentication: A new framework and an implementation," *IEEE Trans. Image Process.*, vol. 15, no. 4, pp. 1042–1049, Apr. 2006.

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

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

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

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

[9] W. Hong, T.-S. Chen, and C.-W. Shiu, "Reversible data hiding for high quality images using modification of prediction errors," *J. Syst. Softw.*, vol. 82, no. 11, pp. 1833–1842, 2009.

[10] 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.

[11] X. Wang, X. Li, B. Yang, and Z. Guo, "Efficient generalized integer transform for reversible watermarking," *IEEE Signal Process. Lett.*, vol. 17, no. 6, pp. 567–570, Jun. 2010.

[12] L. Luo, Z. Chen, M. Chen, X. Zeng, and Z. Xiong, "Reversible image watermarking using interpolation technique," *IEEE Trans. Inf. Forensics Security*, vol. 5, no. 1, pp. 187–193, Mar. 2010.

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

[14] 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.

[15] J. Hwang, J. Kim, and J. Choi, "A reversible watermarking based on histogram shifting," in *Proc. IWDW*, vol. 4283. 2006, pp. 348–361.

[16] F.-H. Hsu, M.-H. Wu, and S.-J. Wang, "Reversible data hiding using side-match predictions on steganographic images," *Multimedia Tools Appl.*, vol. 67, no. 3, pp. 571–591, 2013.

[17] H.-T. Wu and J. Huang, "Reversible image watermarking on prediction errors by efficient histogram modification," *Signal Process.*, vol. 92, no. 12, pp. 3000–3009, 2012.

[18] X. Li, B. Li, B. Yang, and T. Zeng, "General framework to histogram-shifting-based reversible data hiding," *IEEE Trans. Image Process.*, vol. 22, no. 6, pp. 2181–2191, Jun. 2013.

[19] X. Li, W. Zhang, X. Gui, and B. Yang, "A novel reversible data hiding scheme based on two-dimensional difference-histogram modification," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 7, pp. 1091–1100, Jul. 2013.

[20] T. Kalker and F. M. J. Willems, "Capacity bounds and constructions for reversible data-hiding," *Proc. SPIE*, vol. 5020, pp. 604–611, Jun. 2003.

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

[22] S.-J. Lin and W.-H. Chung, "The scalar scheme for reversible information-embedding in gray-scale signals: Capacity evaluation and code constructions," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 4, pp. 1155–1167, Aug. 2012.

[23] B. Chen, W. Zhang, K. Ma, and N. Yu, "Recursive code construction for reversible data hiding in DCT domain," *Multimedia Tools Appl.*, vol. 72, no. 2, pp. 1985–2009, 2014.

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

[25] 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.

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

[27] 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, Feb. 2014.

[28] 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.

[29] 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.

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

[31] 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.

[32] J. Fridrich and R. Du, "Secure steganographic methods for palette images," in *Proc. Int. Workshop Inf. Hiding*, vol. 1768. Sep. 1999, pp. 47–60.

[33] C.-H. Tzeng, Z.-F. Yang, and W.-H. Tsai, "Adaptive data hiding in palette images by color ordering and mapping with security protection," *IEEE Trans. Commun.*, vol. 52, no. 5, pp. 791–800, May 2004.

[34] J.-H. Lee and M.-Y. Wu, "Reversible data-hiding method for palette-based images," *Opt. Eng.*, vol. 47, no. 4, pp. 047008-1–047008-9, 2008.

[35] J. Fridrich, M. Goljan, and R. Du, "Lossless data embedding for all image formats," *Proc. SPIE*, vol. 4675, pp. 572–583, Apr. 2002.

[36] B. Ou, X. Li, and W. Zhang, "PVO-based reversible data hiding for encrypted images," in *Proc. IEEE ChinaSIP*, Jul. 2015, pp. 831–835.

[37] C.-W. Lee and W.-H. Tsai, "A data hiding method based on information sharing via PNG images for applications of color image authentication and metadata embedding," *Signal Process.*, vol. 93, no. 7, pp. 2010–2025, 2013.

[38] C.-W. Lee and W.-H. Tsai, "A secret-sharing-based method for authentication of grayscale document images via the use of the PNG image with a data repair capability," *IEEE Trans. Image Process.*, vol. 21, no. 1, pp. 207–218, Jan. 2012.

[39] H. Wu, H. Wang, H. Zhao, and X. Yu, "Multi-layer assignment steganography using graph-theoretic approach," *Multimedia Tools Appl.*, vol. 74, no. 18, pp. 8171–8196, 2015.

Han-Zhou Wu (S'16) is currently working toward the Ph.D. degree with Southwest Jiaotong University, Chengdu, China.

He is a Visiting Scholar with New Jersey Institute of Technology, Newark, NJ, USA. He specializes in data embedding applications, such as reversible data hiding, steganography and steganalysis, watermarking, and image forensics.

Dr. Wu is a Student Member of the IEEE Signal Processing Society. He was selected to participate in Yahoo! Hack Beijing 2013 based on his technical merit. He once ranked in the 18th place in the Google Cup ACM-ICPC Fudan Invitational Programming Contest with his team members.

Yun-Qing Shi (F'05) received the M.S. degree from Shanghai Jiao Tong University, Shanghai, China, and the Ph.D. degree from University of Pittsburgh, Pittsburgh, PA, USA.

He has been with New Jersey Institute of Technology, Newark, NJ, USA, since 1987. He has authored or co-authored over 300 papers, one book, and five book chapters, and co-edited ten books, three special issues, and 13 proceedings. He holds 30 U.S. patents. His research interests include data hiding, forensics and information assurance, and visual signal processing and communications.

Dr. Shi has been a member of a few IEEE technical committees. He served as the Technical Program Chair of IEEE International Conference on Multimedia and Expo in 2007, a Co-Technical Chair of IEEE Multimedia Signal Processing in 2005, a Co-General Chair of IEEE MMSP02, and a Distinguished Lecturer of IEEE Circuits and Systems Society, and has served as a Co-Technical Chair of the annual International Workshop on Digital Forensics and Watermarking since 2006. He serves as an Associate Editor of IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, and an Editorial Board Member of a few journals. He has also served as an Associate Editor of IEEE TRANSACTIONS ON SIGNAL PROCESSING and IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II.

Hong-Xia Wang received the B.S. degree from Hebei Normal University, Shijiazhuang, China, in 1996, and the M.S. and Ph.D. degrees from University of Electronic Science and Technology of China, Chengdu, China, in 1999 and 2002, respectively.

She pursued post-doctoral research with Shanghai Jiao Tong University, Shanghai, China, from 2002 to 2004 and was a Visiting Scholar with Northern Kentucky University, Highland Heights, KY, USA, from 2013 to 2014. She is currently a Professor with the School of Information Science and Technology, Southwest Jiaotong University, Chengdu. She has authored over 90 research papers in refereed journals and conferences, and holds nine authorized patents. Her research interests include multimedia information security, digital forensics, information hiding, and digital watermarking.

Lin-Na Zhou received the Ph.D. degree from Beijing University of Posts and Telecommunications, Beijing, China, in 2007.

She pursued post-doctoral research with Tsinghua University, Beijing. She is currently a Researcher. She has authored 30 peer-research papers. Her research interests include multimedia information security, digital forensics, and information hiding.