High Capacity Reversible Data Hiding in Encrypted Images by Patch-Level Sparse Representation

Xiaochun Cao, *Senior Member, IEEE*, Ling Du, Xingxing Wei, Dan Meng, *Member, IEEE*, and Xiaojie Guo, *Member, IEEE*

***Abstract*—Reversible data hiding in encrypted images has attracted considerable attention from the communities of privacy security and protection. The success of the previous methods in this area has shown that a superior performance can be achieved by exploiting the redundancy within the image. Specifically, because the pixels in the local structures (like patches or regions) have a strong similarity, they can be heavily compressed, thus resulting in a large hiding room. In this paper, to better explore the correlation between neighbor pixels, we propose to consider the patch-level sparse representation when hiding the secret data. The widely used sparse coding technique has demonstrated that a patch can be linearly represented by some atoms in an over-complete dictionary. As the sparse coding is an approximation solution, the leading residual errors are encoded and self-embedded within the cover image. Furthermore, the learned dictionary is also embedded into the encrypted image. Thanks to the powerful representation of sparse coding, a large vacated room can be achieved, and thus the data hider can embed more secret messages in the encrypted image. Extensive experiments demonstrate that the proposed method significantly outperforms the state-of-the-art methods in terms of the embedding rate and the image quality.**

***Index Terms*—Image encryption, reversible data hiding (RDH), sparse coding.**

Manuscript received September 28, 2014; revised January 19, 2015; accepted April 9, 2015. Date of publication April 30, 2015; date of current version April 13, 2016. This work was supported in part by the National High-Tech Research and Development Program of China under Grant 2014BAK11B03, in part by the National Natural Science Foundation of China under Grant 61422213 and Grant 61332012, in part by the National Basic Research Program of China under Grant 2013CB329305, in part by the 100 Talents Programme of the Chinese Academy of Sciences, and in part by the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant XDA06010701. The work of X. Guo was supported in part by the National Natural Science Foundation of China under Grant 61402467, and in part by the Excellent Young Talent Programme through the Institute Information Engineering, Chinese Academy of Sciences. This paper was recommended by Associate Editor Y. Zhao. *(Corresponding author: Ling Du.)*

X. Cao is with the School of Computer Science and Technology, Tianjin University, Tianjin 300072, China, and also with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China.

L. Du is with the School of Computer Science and Technology, Tianjin University, Tianjin 300072, China, and also with the School of Computer, Shenyang Aerospace University, Shenyang 110136, China (e-mail: duling@tju.edu.cn).

X. Wei is with the School of Computer Science and Technology, Tianjin University, Tianjin 300072, China.

D. Meng and X. Guo are with the Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China.

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/TCYB.2015.2423678

I. Introduction

REVERSIBLE data hiding (RDH) in images aims to exactly recover both the embedded secret information and the original cover image. It has attracted intensive research interests. Military, medical and legal scenarios are its typical examples, in which even a slight distortion is not tolerable. Many RDH algorithms have already been developed, such as image compression-based [1], [2], difference expansion-based [3]–[7], histogram shift (HS)-based [8]–[11], image pixel pair based [12], [13], and dual/multi-image [14], [15] hiding methods. Recently, due to the requirement of privacy protection [16], [17], the cover owner usually encrypts the original content before transferring it to the data manager. Meanwhile, the data manager may want to embed additional messages into the encrypted image for authentication or steganography [18], even though the content of the original image is unknown to him. In this situation, hiding data in the encrypted image is an intuitive and effective way to meet such requirement. To hide data in encrypted domains, some digital watermarking [19], [20]-based schemes are proposed. Besides, the commutative watermarking and ciphering schemes for digital images are introduced in [21] and [22]. Although the methods mentioned above have provided promising performance in encrypted domains, they are insufficient for more sensitive military and medical scenarios, where the image content should be not only kept secret strictly, but also be losslessly recovered after data extraction. Therefore, RDH in encrypted images (RDHEIs) is desirable.

To this end, many RDHEI schemes have been proposed in past years. One of the common techniques is based on manipulating the least-significant-bit (LSB) planes by directly replacing the three LSBs of the cover-image with the message bits, which is kind of the pixel-level compressive methods essentially. In [23], the encrypted image is segmented into a number of nonoverlapped blocks, while each block is divided into two sets. Each block carries one bit by flipping three LSBs of a set for predefined pixels. Hong *et al.* [24] gave an improved version based on [23]. Specifically, they fully harness the pixels in calculating the smoothness of each block and consider the pixel correlations in the border of neighboring blocks. The resulting error rate of extracted-bits is thereby decreased. In [25], the proposed method creates a sparse space to accommodate some additional data by compressing the LSBs of the encrypted image. It is hard to squeeze room by only considering three LSBs of the encrypted images. Instead, Zhang *et al.* [26] chose a half of fourth LSB as

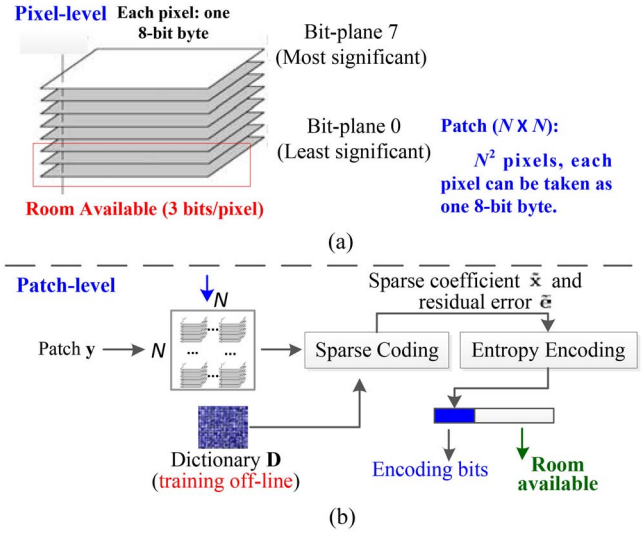

Fig. 1. Comparison between pixel-level and patch-level representation for room preserving. Take one patch with size $N \times N$ for example. (a) For pixel level representation, the number of LSB for one pixel can be set as 3 for better imperceptibility. Therefore, one patch with size $N \times N$ can hide $3N^2$ bits (3 bits per pixel). In contrast, (b) for patch level representation, by using an over-complete dictionary $\mathbf{D}$ (training offline for sparse representation), one patch $\mathbf{y}$ can be represented as a sparse combination of several atoms in the dictionary. That is, $\mathbf{y} = \text{round}(\mathbf{D}\tilde{\mathbf{x}}) + \tilde{\mathbf{e}}$. Thanks to the representation power of sparse coding, only a small number of coefficients $\tilde{\mathbf{x}}$ and residual error $\tilde{\mathbf{e}}$ require space to record. Thus, a higher capacity room is available. For more details, please see our proposed method described in Section III.

the space to carry the data. To further improve the compression ratio, Yin *et al.* [27] selected the smooth blocks in the encrypted image, and embed the additional data into the blocks in a sorted order with respect to block smoothness by using local HS. Although the methods in [23]–[27] divide the image into patches or groups, the preserved spaces are all acquired by using the LSB modification or compression. As the entropy of encrypted images is maximized, it is difficult to losslessly vacate room after encryption (VRAE) using the above methods. To overcome this drawback, the methods of reserving room before encryption (RRBE) are proposed [28], [29]. In [28], a large portion of pixels are utilized to estimate the rest before encryption, the additional data is embedded in the encrypted image by operating the estimating errors. In [29], the reserving room is obtained by embedding LSBs of some pixels into other pixels. The spare space emptied out is three LSBs of the selected pixels. Compared with the VRAE methods, RRBE methods have shown a superior performance.

The success of the above RDHEI methods has verified that the data hiding can be accomplished by exploiting the redundancy within the image. However, for better imperceptibility, only three LSBs can be used for data hiding. As shown in Fig. 1(a), one patch with size $N \times N$ can hide $3N^2$ bits (3 bits per pixel). Actually, for numerous computer vision applications, the image can be analyzed at the patch level rather than at the individual pixel level. Patches contain contextual information and have advantages in terms of computation and generalization. Specifically, because the pixels in certain ranges (like patches or regions) are of strong similarity, the information in any image is correlated in a certain way, especially within a limited local searching range [30]. They can be heavily compressed, and thus can result in a large hiding room. Considering the two aspects, to better exploit the correlations of neighbor pixels, we propose a novel method for high capacity separable reversible data hiding in encrypted images (HC_SRDHEI). As shown in Fig. 1(b), we follow the framework of RRBE and propose to consider the mid-level visual representation with image patches. Using an overcomplete dictionary $\mathbf{D}$ that contains prototype signal atoms, the image patch $\mathbf{y}$ is described by sparse linear combinations of these atoms. That is, only a small number of coefficients $\tilde{\mathbf{x}}$ and the corresponding residual error $\tilde{\mathbf{e}}$ caused by sparse representation, require space to record. Thus, a higher capacity room is available.

Fig. 2 shows the flowchart of the proposed HC_SRDHEI method. For the content owner, the given cover image is represented according to an over-complete dictionary by sparse coefficients. After that, for the given selected patches, the corresponding coefficients and reconstructed residual errors are encoded directly without quantization. For most of the patches, the data size is well reduced in the basis of coefficient representation, thus the vacated room is preserved for high capacity data hiding after image encryption. Note that, for losslessly recovering the cover image, the residual errors are self-embedded into the nonselected patches. The learned dictionary, is also embedded into the encrypted image for further use. At the receiver side, when the receivers accept an encrypted image containing additional data, the processing procedure depends on the role of receiver. If the receiver is a data hider and only has the data hiding key, he can extract the data without knowing the image content. If the receiver is an image owner and only has the encryption key, he can decrypt the image with a better quality. If the receiver is both the image owner and data hider, he has both of two keys in such case. Thus, the data extraction and content recovery are all done, and both results are free of error. In summary, for our proposed framework, the data extraction and image recovery are separable and reversible.

The contributions of this paper are summarized as follows.

1) We present a patch-based RDHEI scheme. Although some other methods also divide the cover image into patches to perform RDHEI, their definition is mainly to consider the correlation of pixels within the patch. Therefore, they are kind of the pixel-level compressive methods essentially. In contrast, we regard the patch as a whole, and represent them using a small number of coefficients, which is beyond the traditional pixel-level case. And thus a high capacity room is available.
2) Our scheme obtains a significant performance improvement over the traditional RDHEI methods. For a certain embedding rate, the visual quality of the directly decrypted image is improved. Meanwhile, for an acceptable peak signal to noise ratio (PSNR), for instance PSNR = 40 dB, the range of embedding rates is greatly enlarged. Experimental results demonstrate that our average maximum embedding rate (MER) reached 1.7 times as large as that of the previous best alternative method conducts.

The remainder of this paper is organized as follows. Section II briefly reviews the related work. The proposed

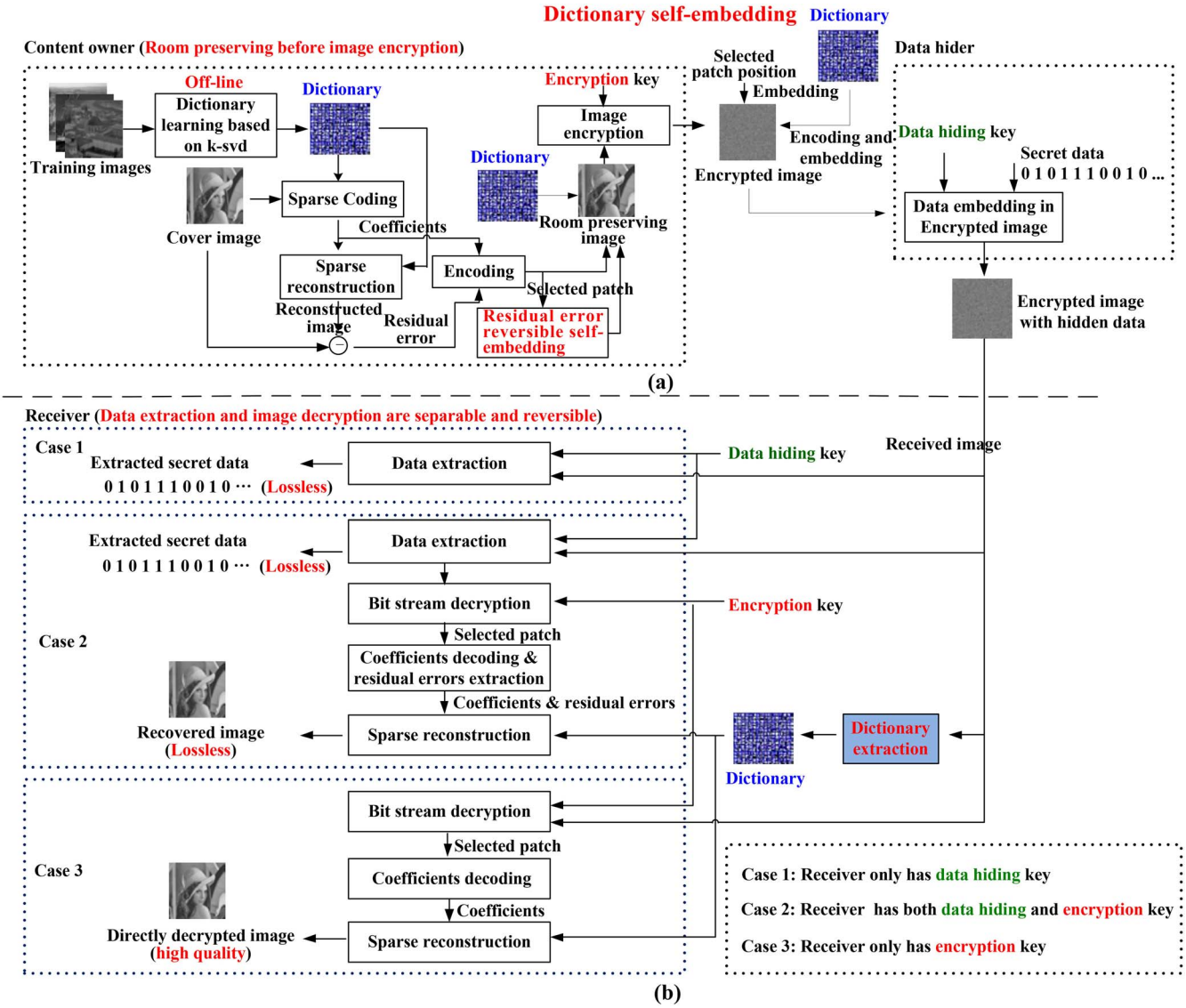

Fig. 2. Flowchart of the proposed HC_SRDHEI method. (a) Sketch of image encryption and data hiding. The content owner encrypts the original image using an encryption key, and the data-hider embed the secret data into encrypted image using a data-hiding key. (b) Sketch of data extraction and image decryption. In the receiver side, we consider three cases depending on whether the receiver has data hiding and/or encryption keys in our framework. Case 1: if a receiver has the data-hiding key, he can extract the data without knowing the image content. Case 2: if the receiver has both of two keys, the data extraction and content recovery are free of error. Case 3: if the receiver has the encryption key, he can decrypt the image with a better quality. Note that there are two keys in our framework. The encryption key is used for encryption, which can be taken as a security measure that turns image into an unreadable cipher. The data hiding key is used to encrypt the hidden data. Anyone who does not possess the data hiding key could not extract the embedded data.

method is detailed in Section III. Section IV reports the extensive experimental results and comparisons to the state-of-the-art methods. Finally, this paper is concluded in Section V.

II. Related Work

Recently, a series of RDHEI schemes have been designed. Generally, these RDHEI methods can be grouped into two categories. The first one is to VRAE, and the second one is to RRBE. For VRAE, some classic methods can be found in [23]–[27]. For RRBE, two major proposed methods are [28] and [29]. Next, we will give a detailed introduction about them.

In [23], the original image cover is first encrypted, and then secret data are embedded by modifying a small proportion of the encrypted image. The receiver first decrypts the encrypted image, and then extracts the embedded data and recovers the original cover image based on the decrypted version. The limitation of this method is that, each block is embedded only one bit payload. Moreover, if the block size is relatively small, the error bits of data extraction increase. To overcome this shortcoming, an improved RDHEI method using side match is proposed by Hong *et al.* [24]. First, it fully exploits pixels when evaluating the smoothness by summing vertical and horizontal differences in image blocks, which leads to a better estimation of the smoothness of blocks for data extraction and image recovery. Second, it adopts the side match technique to concatenate the borders of the recovered blocks to the unrecovered blocks. The error rate obtained by [24] is much lower than [23]. However, another

property for RDHEI, i.e., separability, is not taken into account [23], [24]. That is, the data extraction should be separable from the content decryption. From this point of view, Zhang [25] proposed a novel scheme for separable RDHEIs. The content owner encrypts the original uncompressed image using an encryption key, and then the data hider compresses the LSBs of the encrypted image using a data hiding key to create a sparse space to accommodate some additional data. In the receiver side, three cases that the receiver has encryption and/or data hiding keys are considered. Zhang *et al*. [26] proposed a novel scheme of RDHEIs based on lossless compression of encrypted data by using LDPC code. The data hider compresses a half of the fourth LSB in the cipher-text image, and inserts the compressed data and the additional data into the half of the fourth LSB using efficient embedding method. Moreover, Yin *et al.* [27] proposed a RDHEI scheme which also offers error-free data extraction. The cover image is partitioned into nonoverlapping blocks and multigranularity encryption is applied. The data hider selects the several smoother blocks for data embedding.

Since the entropy of the image has been maximized due to encryption, the room vacating is more difficult than the cover image. Thus the schemes in [23]–[27] can only achieve small payloads even with the advance of compressing encrypted images [31], [32]. The MER in [23]–[27] are all less than 0.2 bits per second. Since losslessly vacating room from encrypted images is relatively difficult and sometimes inefficient, the RRBE methods [28], [29] are proposed. Instead of embedding data in encrypted images directly, Zhang *et al*. [28] proposed to estimate some pixels before encryption, and then the additional data are embedded in the estimating errors. Moreover, Ma *et al.* [29] designed an effective scheme by RRBE. They first empty out room by embedding LSBs of some pixels in one region into another region via a traditional RDH method, and then encrypt the image. As a result, the positions of these LSBs in the first region of encrypted image can be used to hide data. This method can separately extract hidden data and decrypt the image. Moreover, they can embed more than ten times as large payloads as those of previous methods discussed above. However, the spare space emptied out is limited to at most three LSB-planes per pixel. Therefore, the MER is only about 0.5 bits per second in [29].

The proposed HC_RDHEI method inherits the merits of RRBE based on patch level sparse representation. Not only does the proposed method separate the data extraction from image decryption but also achieve excellent performance. Moreover, different from the above methods mainly considering pixel-level compressive property, our scheme takes the patch as a whole, and represents them using sparse coding. As a result, a high capacity is achieved.

III. Proposed Method

In this section, we give a detailed introduction about our HC_SRDHEI in the following three aspects: 1) encrypted image generation; 2) data hiding in the encrypted image; and 3) data extraction and image recovery. For simplicity, we use the grayscale images with 8 bits per pixel. The extension from gray images to color images is straightforward.

A. Encrypted Image Generation

For the image owner, to generate encrypted images, three phases: 1) sparse representation; 2) self-reversible embedding; and 3) stream encryption, are involved. Given a cover image, we first divide it into patches that are then represented according to an overcomplete dictionary via sparse coding. Then, the smoother patches with lower residual errors are selected for room reserving. These selected patches are represented by the sparse coefficients, and the corresponding residual errors are encoded and reversibly embedded into the other nonselected pathes with a standard RDH algorithm. Finally, the room preserved and self-embedded image is encrypted to generate the finally version.

1) Sparse Representation: For reserving room to hide data, we train the dictionary based on K-means singular value decomposition (K-SVD) algorithm [33], [34], which is widely used for designing over-complete dictionaries that lead to sparse signal representation. Note that, the K-SVD training is an offline procedure, and the corresponding dictionary produced by training is then considered fixed for the whole RDH procedure. Given a cover image $\mathbf{I}$ with size $N_1 \times N_2$, we first divide it into a bunch of nonoverlapped $N \times N$ patches. Unless stated, the patch size N is set to 4 as default in our algorithm. Denote S as the number of patches of $\mathbf{I}$, and $S = N_1 \times N_2 / N \times N$. For each patch, the pixel values are vectorized as $\mathbf{y}_i \in \mathbb{R}^{n\times 1}$ ($i = 1, 2, \ldots, S$, and $n = N^2$). Therefore, the image $\mathbf{I} \in \mathbb{R}^{n\times S}$, which contains S column vectors $\{\mathbf{y}_i\}_{i=1}^{S}$. Using an overcomplete dictionary matrix $\mathbf{D} \in \mathbb{R}^{n\times K} (K > n)$ that contains K prototype signal atoms for columns, $\{\mathbf{d}_j\}_{j=1}^{K}$, every image patch $\mathbf{y}_i$ can be represented as a sparse linear combination of these atoms

$$\min_{\mathbf{x}_i} ||\mathbf{y}_i - \mathbf{D}\mathbf{x}_i||_2^2 \quad \text{subject to} \quad ||\mathbf{x}_i||_0 \leq L \tag{1}$$

where $||\cdot||_0$ is the l^0 norm, counting the nonzero entries of a vector. L is a predetermined number of nonzero entries. The coefficient vector $\mathbf{x}_i \in \mathbb{R}^{K\times 1}$ contains the representation coefficients of $\mathbf{y}_i$, and is expected to be sparse. Note that once the well trained dictionary $\mathbf{D}$ is obtained, it can be used for any cover image. The computational complexity analysis for training and the training time will be discussed in Section IV.

Actually, the approximation of $\mathbf{y}_i$ using $\mathbf{D}\mathbf{x}_i$ needs not to be exact, and could absorb a moderate error. In other words, the representation of $\mathbf{y}_i$ may be either exact $\mathbf{y}_i = \mathbf{D}\mathbf{x}_i$ or approximate, $\mathbf{y}_i \approx \mathbf{D}\mathbf{x}_i$. This suggests an approximation that trades off accuracy of representation with its simplicity. Therefore, we make an error correction step for lossless image recovery. Moreover, the sparse coefficients $\mathbf{x}_i$ are adjusted to integers $\tilde{\mathbf{x}}_i = \text{round}(\mathbf{x}_i)$ for the convenience of encoding. Therefore, $\mathbf{y}_i$ can be reconstructed by

$$\mathbf{y}_i = \text{round}\left(\mathbf{D}\tilde{\mathbf{x}}_i\right) + \tilde{\mathbf{e}}_i \tag{2}$$

where $i = 1, 2, \ldots, S$, $\tilde{\mathbf{x}}_i \in \mathbb{Z}^{K\times 1}$, and $\tilde{\mathbf{e}}_i \in \mathbb{Z}^{n\times 1}$. Here, $\tilde{\mathbf{e}}_i$ is considered as the residual error, which contains two parts: 1) the reconstructed error caused by sparse coding and 2) the

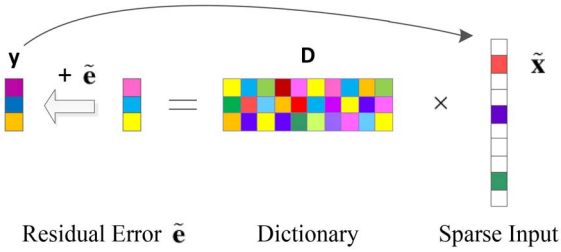

Fig. 3. Sparse representation for each patch based on K-SVD. The corresponding integer sparse coefficient $\tilde{\mathbf{x}}$ and residual error $\tilde{\mathbf{e}}$ are obtained.

rounding error. Note that, at the receiver side, once $\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{e}}_i$ are received, we can exactly recover the image content. The sparse representation for each patch based on K-SVD is shown in Fig. 3.

After error correction, the coefficient $\tilde{\mathbf{x}}_i$ and residual error $\tilde{\mathbf{e}}_i$ are encoded into binary bits. For the sparse coefficient $\tilde{\mathbf{x}}_i$, as most elements in $\tilde{\mathbf{x}}_i$ are zeros, we only encode the L nonzero coefficients. Therefore, $\tilde{\mathbf{x}}_i$ is represented by the indices $\mathbf{p}_i \in \mathbb{Z}^{L\times 1}$ and nonzero coefficients value $\mathbf{v}_i \in \mathbb{Z}^{L\times 1}$. Then for $\mathbf{p}_i$, we convert it into a number of position bits (n_i^p bits). Note that, n_i^p equals to $\lceil \log_2 K \rceil$, where K is the length of dictionary $\mathbf{D}$. As for $\mathbf{v}_i$, we represent it by its corresponding complementary bits, called value bits (n_i^v bits). As the coefficient values mostly fall into the range [−1024,1023], $\mathbf{v}_i$ can be represented by 11 bits, that is to say, $n_i^v = 11$. Once there exist some values out of range, we truncate them into [−1024,1023]. Note that, in this case, the corresponding errors are transferred to $\tilde{\mathbf{e}}_i$, which will not affect the precision of reconstruction. For residual error $\tilde{\mathbf{e}}_i$, we encode them using context-based adaptive variable length coding [35], [36]. The final number of encoded bits for error bits is denoted as n_i^e, and n_i^e is usually different for patches with diverse texture.

2) Self-Reversible Embedding: For room reserving, we select several patches to construct a smoother area $\mathbf{A} \in \mathbb{R}^{n\times C}$ for room reserving, which contains C column vectors $\{\mathbf{y}_k\}_{k=1}^C$. Here, C is the selected patch number, and the size of area $\mathbf{A}$ is nC.

As for smoother patches selection, it is done by computing the length of error bits, and observing which patches are simpler to represent. The patches with smaller error bits are likely smoother than the patches with larger ones. For example, the background, parts of face or parts of clothes will be simpler than the other complex ones such as eyes and hair. To sum up, the patches $\{\mathbf{y}_k\}_{k=1}^C$, are selected according to a binary value of flag bit F_i, which is computed by

$$F_i = \frac{\text{sign}\left(n_i^e - \delta\right) + 1}{2}. \tag{3}$$

In our scheme, $F_i = 0$ indicates the corresponding patch is selected for room persevering. Moreover, δ is the threshold for selection, and is determined by

$$\delta = \mathbf{q}_C, \quad \mathbf{q} = \begin{bmatrix} n^e_{\varphi(1)} \\ n^e_{\varphi(2)} \\ \cdots \\ n^e_{\varphi(S)} \end{bmatrix} \tag{4}$$

where C is the selected patch number, $\mathbf{q}$ is the result of sorting n_i^e in the ascending order, $i, \varphi(i) = 1, 2, \ldots, S$. Note that, the threshold δ for smoother area selection is dependent on the cover image, and is different for smooth and complex images.

For each selected patch $\{\mathbf{y}_k\}_{k=1}^C$, except for sparse coefficients (position bits n_k^p and value bits n_k^v), we still need another parameter to imply the locations of the next selected smoother patches for data hider. This parameter occupies another n_k^b bits, called parameter bits. Note that, the position of the first selected patch is finally embedded in the encrypted image by RDH algorithm. The positions ($\mathbf{p}_{x_k}$, $\mathbf{p}_{y_k}$) of the selected patches $\{\mathbf{y}_k\}_{k=1}^C$ in the original image are computed by

$$\mathbf{p}_{x_k} = \begin{cases} \dfrac{p_k}{N_2/N}, & \text{mod}(p_k, N_2/N) = 0 \\ \text{floor}\left(\dfrac{p_k}{N_2/N}\right) + 1, & \text{mod}(p_k, N_2/N) \neq 0 \end{cases} \tag{5}$$

$$\mathbf{p}_{y_k} = \begin{cases} N_2/N, & \text{mod}(p_k, N_2/N) = 0 \\ \text{mod}(p_k, N_2/N), & \text{mod}(p_k, N_2/N) \neq 0 \end{cases} \tag{6}$$

where p_k is the patch index in the original image which is scanned based on raster-scan order. As discussed above, parameter bits occupy $n_k^b = \lceil \log_2 N_1/N \rceil + \lceil \log_2 N_2/N \rceil$ bits for each patch. For the dictionary, it is also needed to be encoded and embedded into the encrypted image, which occupy another n^a bits totally, and each selected patch is assigned $\tilde{n}_k^a = n^a/C$ bits. Therefore, as shown in Fig. 4, each selected patch can provide $n_k^d = 8N^2 - L(n_k^p + n_k^v) - n_k^b - \tilde{n}_k^a$ bits for data hiding. Specifically, in the most state-of-the-art RRBE method [29], the spare space emptied out is limited to at most three LSB-planes. That is, $3N^2$ bits for each patch. As we regard the patch as a whole, and represent them using a small number of coefficients, which is beyond the traditional pixel-level case. In other words, n_k^d is usually larger than $3N^2$. For more detail results, please refer to Section IV.

According to the discussion above, for each selected patch $\{\mathbf{y}_k\}_{k=1}^C$, we have $n_k^p = \lceil \log_2 K \rceil$, $n_k^v = 11$ for empirically setting, $n_k^b = \lceil \log_2 N_1/N \rceil + \lceil \log_2 N_2/N \rceil$. As to $\tilde{n}_k^a$, we have $\tilde{n}_k^a = n^a/C$. Table I shows the size of n^a with different parameters K and L setting. K is the length of dictionary $\mathbf{D}$ and L is a predetermined number of nonzero entries in sparse coefficient. It is noteworthy that n^a is only dependent on the training set and parameters K and L setting. The parameters learning for K and L will be discussed latter in Section IV. Therefore, for a certain image, once the dictionary is determined, n_k^p, n_k^v, n_k^b, $\tilde{n}_k^a$, and n_k^d is identical for each patch, we denote them as n^p, n^v, n^b, $\tilde{n}^a$, and n^d, respectively, for simpleness.

Assume the size of the data to be hidden is denoted as M, the relationship between selected patch number C, data hidden size M and room preserving per patch n^d is

$$C = \frac{M}{n^d} = \left\lceil \frac{M}{8N^2 - L(n^p + n^v) - n^b - \tilde{n}^a} \right\rceil. \tag{7}$$

Then, we rewrite (7) as

$$C = \left\lceil \frac{M + n^a}{8N^2 - U} \right\rceil \tag{8}$$

where

$$U = L\left(\lceil \log_2 K \rceil + 11\right) + \left(\left\lceil \log_2 \frac{N_1}{N} \right\rceil + \left\lceil \log_2 \frac{N_2}{N} \right\rceil\right). \tag{9}$$

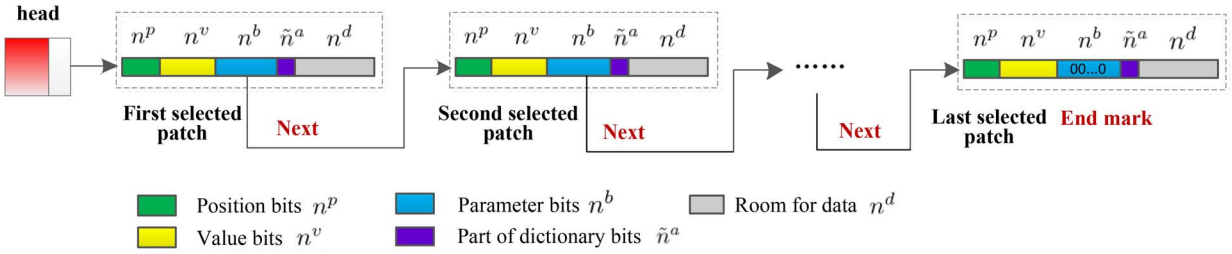

Fig. 4. Structure for selected patches in one cover image. It follows the chain structure between patches. The positions of the selected patches (parameter bits) are embedded after stream encryption patch by patch to inform the data hider where they can embed. The end mark (n^b zeros) are set in the last selected patch. Note that: 1) the position of the first selected patch and vacated room size per patch are finally embedded in the encrypted image by RDH algorithm; 2) all rooms for patches are collected together to do data hiding and only one dictionary for a cover image is required; and 3) n^d is identical for each patch.

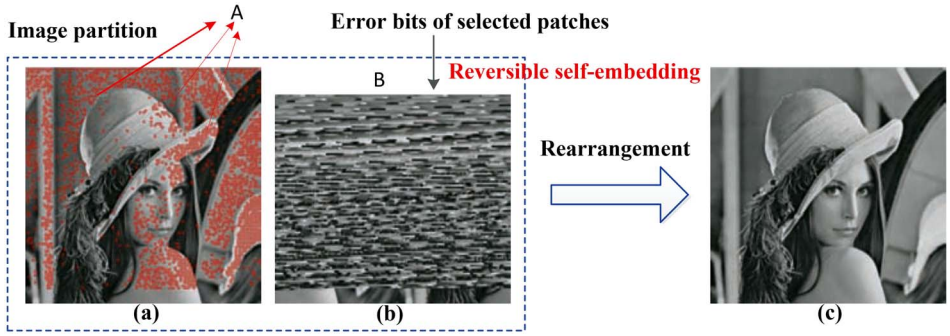

Fig. 5. Illustration of image partition and self-embedding (image size: 512 × 512, patch size: 4 × 4, and embedding rate = 0.75 bits per second). (a) Patches denoted with red squares are selected to construct **A**. (b) Nonselected patches are rearranged into area **B**. (c) Room preserved and self-embedded image.

TABLE I
DICTIONARY ENCODING SIZE (n^a) WITH DIFFERENT PARAMETERS K AND L SETTING (BIT)

K	32	64	128	256	512	1024	2048
L=2	6728	13424	26824	53344	106168	211232	419184
L=3	6552	13304	26152	52448	104288	207760	413480

Finally, these selected patches $\{\mathbf{y}_k\}_{k=1}^{C}$ are represented by the sparse coefficients. The room for parameter bits, dictionary bits and hidden data are filled with random bits. The parameter bits and dictionary bits are set after encryption, and the vacated room is preserved for data hider. For lossless image recovery, the corresponding residual errors $\{\tilde{\mathbf{e}}_k\}_{k=1}^{C}$ are reversibly embedded into the other nonselected pathes, which construct the area **B**, with a standard RDH algorithm [37]. For simplicity, the size of area **B** is computed by $N_{B_1} \times N_{B_2}$, where

$$N_{B_1} = N \times \text{floor}\left(\frac{S-C}{N_2/N}\right)$$
$$N_{B_2} = N_2. \tag{10}$$

Thus, the cover image is converted into its room preserved and self-embedded version $\mathbf{I}_c$. The illustration of image partition and reversible self-embedding process is shown in Fig. 5. Note that, this step does not rely on any specific RDH algorithm.

3) Image Encryption: For the room preserved self-embedded image $\mathbf{I}_c$, we generate the encrypted image $\mathbf{I}_e$ by a stream cipher, such as RC4 or data encryption standard in cipher feedback mode [38]. Denote the eight bits of the pixel $p_{i,j}(i = 1, 2, \ldots N_1, j = 1, 2, \ldots N_2)$ as $b_{i,j,0}$, $b_{i,j,1}$, $b_{i,j,2}$, $b_{i,j,3}$, $b_{i,j,4}$, $b_{i,j,5}$, $b_{i,j,6}$, and $b_{i,j,7}$. Thus

$$b_{i,j,k} = \lfloor p_{i,j}/2^m \rfloor \bmod 2, \quad m = 0, 1, \ldots, 7. \tag{11}$$

Then, the encrypted bit stream can be expressed as

$$b'_{i,j,m} = b_{i,j,m} \oplus r_{i,j,m}, \quad m = 0, 1, \ldots, 7 \tag{12}$$

where $r_{i,j,m}$ is a pseudo-random bit generated by the encryption key K_e using the stream cipher.

After encryption, we set the parameter bits in the selected patches $\{\mathbf{y}_k\}_{k=1}^{C-1}$ to inform the data hider the positions of the next patches $\{\mathbf{y}_k\}_{k=2}^{C}$ they can embed. As shown in Fig. 4, the data hiders are able to access each embedded patch by traversing this list structure. Note that, the end mark is set in the last selected patch $\mathbf{y}_C$ with n^b bits zeros. As for the well trained dictionary, its corresponding encoding bits n^a, are encrypted with K_e and also embedded into the corresponding reserving room. After that, the position of the first selected patch $\mathbf{y}_1$ and the vacated room size n^d for data hiding, are also embedded by RDH algorithm proposed in [37]. Here, the position is encrypted and can be decrypted either by encryption key K_e or data hiding key K_d. In contrast, the data hiding size n^d is encrypted and can only be decrypted by data hiding key K_d. Finally, the encrypted image $\mathbf{I}_e$ is obtained.

B. Data Hiding in Encrypted Images

Once the encrypted image is received, the data hider can embed secret data for management or authentication requirement. The embedding process starts with locating the encrypted version of area **A**. Since the image owner has embedded the position of the first room preserving patch and the room size for each patch in the encrypted image, it is effortless for the data hider to know where and how many bits they can modify. After that, the data hider scans each selected

patch in the encrypted image $\mathbf{I}_e$, and simply makes use of bit replacement to substitute the corresponding bits reserved for secret data. Here, we assume the selected patch number is denoted as C, our MER for the data hider is computed as follows:

$$\text{MER} = \frac{C \times \left(8N^2 - L(n^p + n^v) - n^b\right) - n^a}{N_1 \times N_2} \tag{13}$$

where n^a is the dictionary size and is fixed for our algorithm. After data hiding, the position of the first data hiding patch and the hiding room size for each patch are also embedded into the encrypted image containing additional embedded data with RDH algorithm. Note that, the secret data is encrypted according to the data hiding key K_d before hiding.

C. Data Extraction and Image Recovery

With the encrypted image containing additional embedded data, the receiver faces three situations depending on whether the receiver has data hiding and/or encryption keys. The data extraction and image decryption can be processed separately.

1) Data Extraction With Only Data Hiding Key: For the receiver who only has data hiding key K_d, he first extracts and computes the starting position and the hiding room size for each patch and divides the received image into nonoverlapped $N \times N$ patches. Then, data extraction is finished by checking the last n^d bits for the selected patches in the received image. After that, all original hidden data are extracted and recovered with the data hiding key K_d. The extracted data is lossless. Moreover, the receiver does not access to image content in the entire extraction process.

2) Image Decryption With Only Encryption Key: In this case, the receiver has the encryption key K_e only. After extraction the position of the first selected patch by RDH algorithm, all the selected patches are identified one by one. Moreover, the dictionary $\mathbf{D}$ is also obtained by extraction. After patch segmentation of the received image, the decryption procedure is performed and it includes two cases: 1) unselected patch decryption and 2) selected patch decryption. For unselected patch, the content can be directly decrypted according to the encryption key

$$b_{i,j,m} = b'_{i,j,m} \oplus r_{i,j,m}, \quad m = 0, 1, \ldots, 7 \tag{14}$$

where $r_{i,j,m}$ is the pseudo-random bit generated by the encryption key K_e. $b'_{i,j,m}$ and $b_{i,j,m}$ are the encrypted bit and the decrypted bit for the pixel $p_{i,j}$, respectively. Consequently, the unselected patch decryption is losslessly achieved. For the selected patch, we first decrypt the encoded bits by (14) based on encrypting K_e. Then, the position and value of the sparse coefficients for each selected patch $\{\mathbf{y}_k\}_{k=1}^{C}$ are determined. After that, the corresponding coefficient, denoted as $\tilde{\mathbf{x}_k}$, are obtained. Then, the decrypted patch $\mathbf{y}_\mathbf{k}^\mathbf{d}$ is computed via

$$\mathbf{y}_\mathbf{k}^\mathbf{d} = \text{round}\left(\mathbf{D}\tilde{\mathbf{x}}_\mathbf{k}\right) \tag{15}$$

where $k = 1, 2, \ldots, C$, $\mathbf{D}$ is the trained dictionary. Since both the unselected and selected patches are decrypted, the image decryption in our proposed method is completed.

TABLE II
TRAINING TIME WITH DIFFERENT PARAMETERS K AND L SETTING (HOURS)

K	32	64	128	256	512	1024	2048
L=2	2.36	3.52	2.43	2.10	2.32	2.91	3.55
L=3	3.10	3.07	3.14	3.25	3.50	4.29	5.29

3) Data Extraction and Image Recovery With Both Data Hiding and Encryption Keys: If the receiver has both the data hiding key K_d and encryption key K_e, the data extraction and image recovery achieve full reversibility. On the one hand, with the data hiding key K_d, one can extract the hidden secret data without any error. On the other hand, with the encryption key K_e, they first perform directly image decryption, then the corresponding coefficient for selected patches $\{\mathbf{y}_k\}_{k=1}^{C}$, denoted as $\tilde{\mathbf{x}_k}$, are obtained. After that, the residual errors $\tilde{\mathbf{e}_k}$, are extracted from the nonselected patches (corresponding to area $\mathbf{B}$). Therefore, the recovery patch $\mathbf{y}_\mathbf{k}^\mathbf{r}$ is computed as

$$\mathbf{y}_\mathbf{k}^\mathbf{r} = \text{round}\left(\mathbf{D}\tilde{\mathbf{x}}_\mathbf{k}\right) + \tilde{\mathbf{e}}_\mathbf{k} \tag{16}$$

where $k = 1, 2, \ldots, C$. As the patch recovery is based on the lossless coefficients and residual errors, there exists no errors for the selected patches. Moreover, thanks to the RDH algorithm, the nonselected patches are also recovered losslessly after residual errors extraction. That is to say, the image recovery in our proposed method is free of any error.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we conduct several experiments to evaluate the proposed algorithm, which include: choice of dictionary parameters, image encoding, and performance analysis on public available standard images. After that, the particular comparisons between our method and the most state-of-the-art competitor [29] on Kodak,[1] BOSSBase [39], PASCAL [40], and Holidays [41] are described. Finally, we make the corresponding computational complexity analysis.

A. Choice of Dictionary Parameters

Our dictionary training is based on 786 432 patches with size 4×4 taken from 48 standard 8-bit grayscale images in the University of Southern California, Signal and Image Processing Institute image database. For the training process, we adopt K-SVD as the trainer. In our implementation, the maximal number of iterations, T, of K-SVD is set to 50. The experiment PC configuration is: Intel i7-2600, CPU 3.40 GHz, 10 GB RAM. The training time for different K and L is shown in Table II. The output dictionary has the size of $16 \times K$. The coefficients are computed using orthogonal matching pursuit (OMP) with a fixed maximal number of nonzero coefficient L. K and L are selected by referring to the performance comparison of our algorithm. For making a best choice of dictionary parameters which represent the higher embedding rate, we compute the average position bits ($\bar{n}^p$), value bits ($\bar{n}^v$), and error bits ($\bar{n}^e$) for all the patches in the training set. The results is shown in Table III. Note that, to reduce the effect of random samples, the results shown in Table IV

[1] http://www.r0k.us/graphics/kadak

TABLE III
PERFORMANCE COMPARISON WITH DIFFERENT PARAMETERS (K, L) IN K-SVD DICTIONARY TRAINING

	Encoding Entry	K=32	K=64	K=128	K=256	K=512	K=1024	K=2048	Average
L=2	*Position bits* ($\bar{n}_p$)	10	12	14	16	18	20	22	16
	Value bits ($\bar{n}_v$)	22	22	22	22	22	22	22	22
	Error bits ($\bar{n}_e$)	71.43	68.09	66.41	64.79	64.14	61.89	62.74	65.64
	Total bits	103.43	**102.09**	102.41	102.79	104.14	103.89	106.74	**103.64**
L=3	*Position bit* ($\bar{n}_p$)	15	18	21	24	27	30	33	24
	Value bit ($\bar{n}_v$)	33	33	33	33	33	33	33	33
	Error bit ($\bar{n}_e$)	60.42	56.26	53.46	51.05	49.66	47.02	44.69	51.79
	Total bits	108.42	**107.26**	107.46	108.05	109.66	110.02	110.69	**108.79**

TABLE IV
PSNR IN DIRECTLY DECRYPTED IMAGES (DB) WITH DIFFERENT ER FOR THE KODAK IMAGE DATABASE

Ma [29]	kodim01	kodim02	kodim03	kodim04	kodim05	kodim06	kodim07	kodim08	kodim09	kodim10	kodim11	kodim12
0.1 bpp	50.10	53.22	55.39	53.27	50.67	52.28	54.86	49.64	54.38	54.01	53.37	53.46
0.2 bpp	45.44	50.54	51.98	50.23	45.72	49.58	51.89	43.27	50.97	50.67	50.10	51.22
0.3 bpp	41.08	47.87	48.84	46.90	42.49	44.39	48.71	39.71	47.86	47.48	45.27	47.99
0.4 bpp	37.70	44.83	47.04	44.43	38.81	41.62	46.85	34.42	45.33	44.92	42.79	45.30
0.5 bpp	34.55	43.05	44.71	41.50	35.52	38.75	44.63	29.87	43.19	42.12	39.79	43.33
0.6 bpp	31.12	40.49	41.21	39.18	31.78	36.19	42.15	—	40.40	40.06	36.62	40.92
0.7 bpp	27.95	37.85	38.74	35.79	—	32.33	39.60	—	37.66	37.33	33.29	38.13
0.8 bpp	23.75	34.52	35.60	32.89	—	—	36.80	—	35.16	34.33	29.96	34.28
Average	36.46	44.05	45.44	43.02	30.62	36.89	45.68	**24.61**	44.37	43.87	41.40	44.33
Proposed	**kodim1**	**kodim2**	**kodim3**	**kodim4**	**kodim5**	**kodim6**	**kodim7**	**kodim8**	**kodim9**	**kodim10**	**kodim11**	**kodim12**
0.1 bpp	51.03	54.71	59.22	53.20	49.50	59.60	57.25	48.58	55.98	54.59	59.20	56.18
0.2 bpp	47.92	52.77	55.76	50.52	46.65	51.32	53.80	43.22	52.85	51.93	54.55	52.89
0.3 bpp	44.06	50.72	53.75	48.57	42.73	50.81	52.06	38.76	50.49	49.92	51.58	51.19
0.4 bpp	40.68	48.96	52.11	47.26	39.81	49.25	50.76	34.60	49.26	48.49	50.16	49.97
0.5 bpp	37.25	47.93	50.84	45.14	37.56	46.76	49.54	—	47.80	46.71	48.21	48.61
0.6 bpp	34.75	46.28	49.82	43.50	34.20	44.04	48.53	—	45.28	44.89	46.12	46.91
0.7 bpp	31.98	44.46	48.80	41.50	31.44	42.01	46.56	—	43.94	44.21	43.74	45.28
0.8 bpp	29.36	43.86	48.01	39.95	28.12	39.80	45.10	—	43.20	41.96	41.37	43.65
Average	**39.63**	**48.65**	**52.29**	**46.20**	**38.73**	**47.95**	**50.45**	20.65	**48.63**	**47.84**	**49.36**	**49.33**
Ma [29]	**kodim13**	**kodim14**	**kodim15**	**kodim16**	**kodim17**	**kodim18**	**kodim19**	**kodim20**	**kodim21**	**kodim22**	**kodim23**	**kodim24**
0.1 bpp	47.53	51.04	54.03	55.52	53.82	51.01	54.33	—	54.61	52.25	55.13	52.55
0.2 bpp	40.85	46.61	50.94	51.34	50.93	45.98	50.45	—	50.64	49.43	52.00	49.26
0.3 bpp	35.74	43.21	48.07	47.86	47.71	42.94	46.04	—	46.92	45.33	48.72	43.97
0.4 bpp	30.23	40.07	44.73	45.08	44.76	39.49	43.31	—	44.27	42.64	47.07	39.93
0.5 bpp	—	36.92	41.39	43.20	42.01	36.46	40.37	—	41.34	40.26	44.80	36.52
0.6 bpp	—	33.91	38.19	40.37	40.39	33.65	37.67	—	38.66	38.17	42.42	36.40
0.7 bpp	—	30.66	—	38.38	37.27	30.37	35.43	—	36.58	35.04	40.01	33.13
0.8 bpp	—	27.06	—	35.66	34.07	—	32.70	—	34.10	32.26	37.33	30.27
Average	19.29	38.68	34.67	44.68	43.87	34.99	42.54	—	43.40	41.92	45.94	40.26
Proposed	**kodim13**	**kodim14**	**kodim15**	**kodim16**	**kodim17**	**kodim18**	**kodim19**	**kodim20**	**kodim21**	**kodim22**	**kodim23**	**kodim24**
0.1 bpp	47.22	49.71	57.62	58.13	54.62	50.01	53.65	62.77	56.39	52.61	57.68	56.07
0.2 bpp	40.30	46.57	54.72	54.74	51.34	47.14	49.81	60.36	52.87	50.52	53.91	52.84
0.3 bpp	35.96	43.75	52.72	51.37	49.55	44.15	48.12	60.71	50.30	48.37	51.99	49.93
0.4 bpp	31.61	40.99	50.96	50.06	48.08	41.99	45.72	59.50	49.10	46.13	50.82	47.99
0.5 bpp	26.64	38.34	49.52	48.47	46.51	40.06	43.81	57.91	46.38	44.23	49.31	44.18
0.6 bpp	—	35.83	47.00	46.37	45.55	37.52	42.28	55.95	44.40	42.07	48.23	41.64
0.7 bpp	—	33.79	45.12	44.82	43.63	35.29	40.65	54.38	42.97	40.89	46.04	38.35
0.8 bpp	—	31.51	44.01	43.28	41.03	32.91	38.49	52.85	41.24	38.57	44.67	35.50
Average	**22.72**	**40.06**	**50.21**	**49.65**	**47.54**	**41.13**	**45.32**	**58.05**	**47.96**	**45.42**	**50.33**	**45.81**

are computed as the average over all 786 432 patches in the training set.

From Table III, we can see that, as L increases, the average total bits become larger due to sizable position and value bits. Actually, if L increases from 2 to 3, the average total bits increases from 103.64 bits to 108.79 bits. Therefore, the parameter L is set to 2 undoubtedly. As for another parameter K, according to sparse coding, a larger K leads to a better reconstruction performance, which implies a smaller length to encode residual errors. However, as the parameter K determines the length of the position encoding ($\lceil \log_2 K \rceil$), the larger K also increases the encoding length of coefficients. In accordance with Table III, $K = 64$ and $L = 2$ provide the best performance. Unless stated, the two parameters K and L are set to 64 and 2 as default in our implementation. The resulting dictionary is shown in Fig. 6.

B. Image Encoding

Once the well-trained dictionary is obtained, the given image can be represented by the sparse coding according to this dictionary. For sparse representation, Fig. 7 shows the encoding results of residual errors for each patch. The textures for these four images are from smooth to complex. The pixels in Fig. 7 are corresponded to the length of the encoding residual errors for the patch. The brighter the pixel shown in Fig. 7,

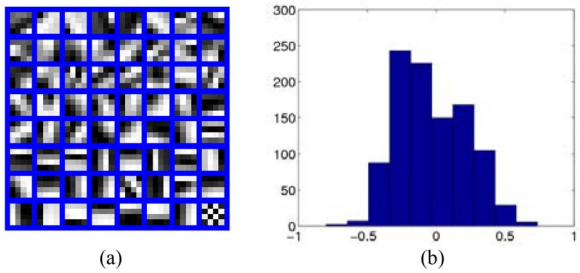

Fig. 6. Demonstration of dictionary. (a) Dictionary obtained by K-SVD with parameters $K = 64$ and $L = 2$. Here, the dictionary atoms are displayed as blocks 4×4. (b) Histogram of the values in the dictionary of (a).

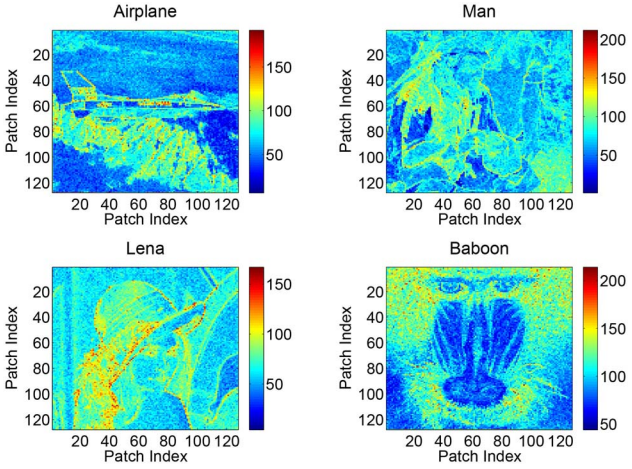

Fig. 7. Visualization of residual error data length for the each patch (dictionary parameters $K = 64$ and $L = 2$). The textures for these four images (*Airplane*, *Man*, *Lena*, and *Baboon*) range from smooth to complex.

the more residual errors of that patch caused by sparse representation. Our encoding strategy allows us to learn which part of the image or which type of image can be easily represented by sparse coding. Clearly, patches with smoother textures, such as backgrounds and clothes, are simpler to represent than complex ones. The patches containing higher frequency elements, such as the edge in image *Airplane* and the hair of image *Lena*, have the higher residual errors. According to the embedding requirement, some patches are selected for data hiding, and its corresponding residual errors are needed to be self-embedded. Fig. 8 shows the selected patches for ER = 0.1051, 0.2613, and 0.5738 bits per second, all of which are marked by red squares. Finally, all of the selected patches are combined into area **A** for data hiding.

C. Reversible Data Hiding

Once the data hiders acquire the encrypted images, they can embed some data for some particular demands. Fig. 9 shows the results of our proposed method with ER = 0.95 bits per second. Fig. 9(a) and (b) shows the original and the encrypted image with embedded data. For the receivers that only have data hiding key K_d, they can extract the hidden data losslessly. Moreover, the receiver with encryption key K_e can directly decrypt the image with higher quality. When both of the keys K_d and K_e are obtained, one can losslessly recover the

Fig. 8. Selected patches demonstration (marked by red squares) for data hiding of proposed method with different ERs.

Fig. 9. Results demonstration of proposed method with ER = 0.95 bits per second. (a) Original image. (b) Encrypted image. (c) Directly decrypted image (PSNR = 48.2584 dB). (d) Recovery image.

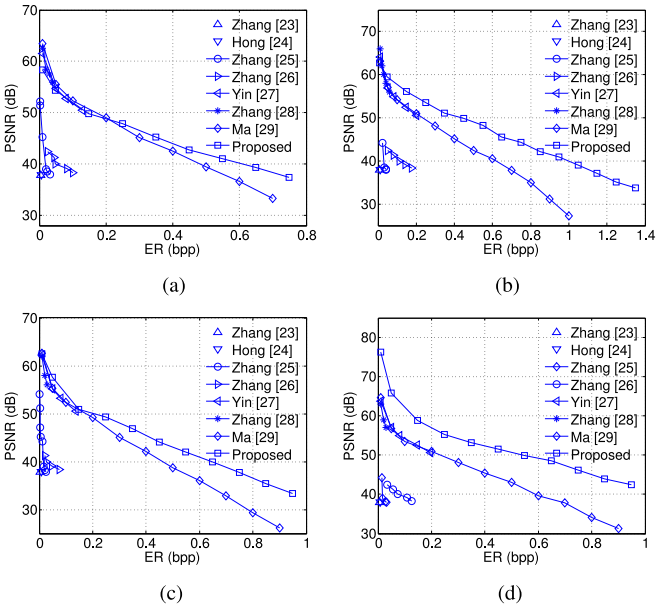

Fig. 10. PSNR comparison for the directly decrypted images with the methods [23]–[29]. (a) *Lena*. (b) *Airplane*. (c) *Man*. (d) *Crowd*.

original images by decrypting and decoding the corresponding reconstructed coefficients and residual errors. The directly decrypted image (PSNR = 48.2584 dB) and recovery image are shown in Fig. 9(c) and (d). Just as it shown, the recovery version is identical to the original image visually, with higher image quality.

To quantitatively measure the performance of our proposed method, we compare our HC_RDHEI algorithm with other conventional schemes. Fig. 10 shows the comparison results with the methods [23]–[29]. As shown in Fig. 10, the ER of our method are mostly better than the state-of-the-art RDH algorithm in encrypted images. Given the ER range that the methods in [23]–[28] can achieve, the gains of PSNR is significantly high. In other aspects, for the acceptable PSNR, such as PSNR $\geqslant$ 40 dB, the range of ER is much wider. As for the

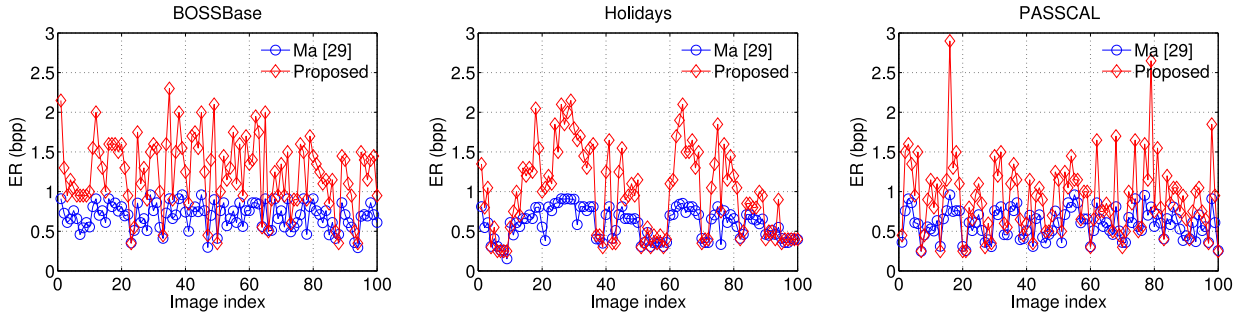

Fig. 11. ER comparison with the closest competitor [29] (PSNR = 40 dB) on 100 images randomly selected from three image datasets, i.e., BOSSBase [39], PASCAL [40], and Holidays [41], respectively. The image indices are obtained based on sorting according to image names.

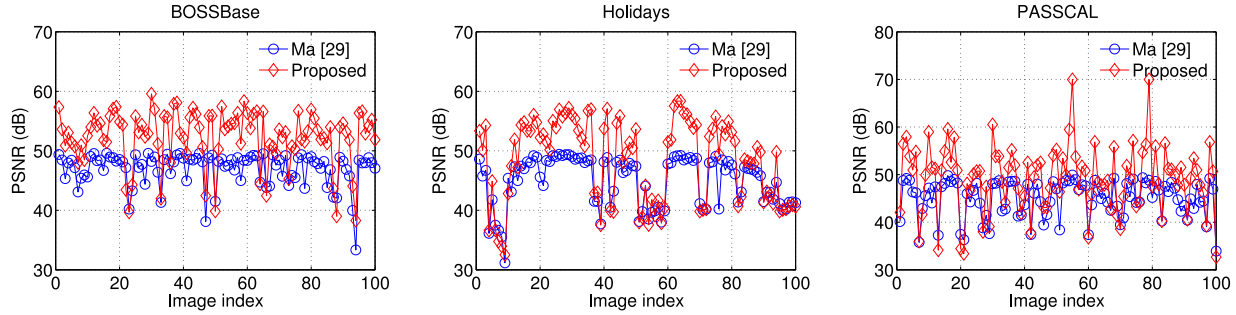

Fig. 12. PSNR comparison with the closest competitor [29] (ER = 0.35 bits per second) on 100 images randomly selected from three image datasets, i.e., BOSSBase [39], PASCAL [40], and Holidays [41], respectively. The image indices are obtained based on sorting according to image names.

four images *Lena*, *Airplane*, *Man*, and *Crowd*, the MERs of our method are 0.61, 1.0, 0.63, and 1.01 bits per second, while the closest competitor [29] has the MERs of 0.49, 0.63, 0.46, and 0.58 bits per second. The performance analysis implies that our proposed method has a very good hiding capacity. That is mainly because that, in the RRBE method [29], the spare space emptied out is limited to at most three LSB-planes ($3N^2$ bits for each patch). For our scheme, if an image with size 512×512, patch size 4×4, and dictionary parameters $K = 64$ and $L = 2$, the hiding bits per patch for this image can give more than 55 bits according to experiment results, while only 48 bits can be preserved in [29].

D. Performance Analysis on Datasets

To evaluate the average performance of our proposed method, the performance analysis on image datasets, i.e., Kodak, BOSSBase, PASCAL, and Holidays are involved. Kodak image database contains 25 color images sized 512×768 or 768×512. For testing, the color images are transformed into gray-level sized 512×512. The comparisons of PSNR in directly decrypted images with different ER are shown in Table IV. Just as shown, the results are mostly better than the most state-of-the-art competitor [29]. For the other three database, in order to clearly demonstrate the performance, we randomly select 100 images from each dataset as the test sets. The images in BOSSBase are gray-levels with size 512×512. For comparison without loss of generality, we also transform these color images in PASCAL and Holidays into gray levels with the same size. The experimental results of comparison with the state-of-the-art method proposed in [29] are shown in Figs. 11 and 12. Actually, for RDHEI method in [29] and ours, the ERs of both methods vary for different images. Given PSNR = 40 dB, as shown in Fig. 11, our algorithm has a much wider range of the embedding rate. The average ER of our method for these three datasets are 1.2458, 0.9873, and 0.9258 bits per second, respectively. As for the method proposed in [29], the average ER are 0.6916, 0.5930, and 0.5925 bits per second for the acceptable directly decrypted image quality (PSNR = 40 dB). The experimental results show that our proposed method reaches about 1.7 times as large payloads as the method in [29]. Meanwhile, given ER = 0.35 bits per second, our method obtains better performance without surprise, as shown in Fig. 12. The average gains of PSNR for three dataset are 5.2963, 3.4312, and 4.2360 dB, respectively. Therefore, we can conclude the summarization that the proposed scheme is more suitable for the RDH applications in encrypted images thanks to its higher hiding capacity and better image quality for direct decryption.

E. Computational Complexity Analysis

For the computational complexity analysis, we mainly consider the three steps: 1) smooth area selection; 2) self-reversible embedding; and 3) image encryption, for our proposed algorithm and the most state-of-the-art competitor [29]. Let the gray-scale image with its size $N_1 \times N_2$, $\tilde{N} = \max(N_1, N_2)$. For smooth area selection, the computational complexity for [29] can be denoted by $O(\tilde{N}^2) + O(\tilde{N} \log \tilde{N})$. The first item represents pixels error computing, the second item represents sorting for smooth area selection. In assessing computational complexities for smooth area selection of our method, we should separately discuss the dictionary training and our proposed data hiding algorithm

for encrypted image. The training process consists of two stages: sparse coding and dictionary update, which are executed iteratively. As mentioned in [42], both stages can be done efficiently in $O(P/NTNKLS') = O(\tilde{N}^2TKLS')$, where P is the number of pixels in the image ($P = N_1 \times N_2$), N is the size of patch, T is the number of iterations, K is the number of atoms in the dictionary, L is the number of nonzero elements in each coefficient vector, and S' is the number of examples in the training set. It is important to note that the K-SVD training process is an offline procedure, and the produced dictionary is then fixed for the whole procedure of proposed method. Practically, the training procedure has been done with a nonoptimized MATLAB code on a regular PC. This procedure can be preprocessed by any user on the PC with common configurations. As for our proposed scheme, the computational complexity mainly depends on sparse representation (encoding) and selected patch recovery used for data hiding (decoding). The encoding adopts the OMP algorithm, the complexity of which is $O(P/NNKL) = O(\tilde{N}^2KL)$. Moreover, we also need sort function for final smooth area selection, Therefore, the computational complexity of our method is $O(\tilde{N}^2KL) + O(2\tilde{N}^2 \log \tilde{N})$. For the other two steps: self-reversible embedding and image encryption, both of [29] and ours adopt traditional RDH algorithm [37] and stream encryption. The computational complexity is nearly the same, which are $O(R\tilde{N}^2)$ and $O(\tilde{N}^2)$ practically, where R is the number of embedding rounds. Therefore, the overall computational complexity mainly depend on smooth area selection, $O(\tilde{N}^2) + O(\tilde{N} \log \tilde{N})$ for the method in [29], and $O(\tilde{N}^2KL) + O(2\tilde{N}^2 \log \tilde{N})$ for ours.

V. Conclusion

This paper has proposed a novel method called the HC_SRDHEI, which inherits the merits of RRBE, and the separability property of RDH methods in encrypted images. Compared to state-of-the-art alternatives, the room vacated for data hiding by our method is much larger used. The data hider simply adopts the pixel replacement to substitute the available room with additional secret data. The data extraction and cover image recovery are separable, and are free of any error. Experimental results on three datasets have demonstrated that our average MER can reach 1.7 times as large as the previous best alternative method provides. The performance analysis implies that our proposed method has a very good potential for practical applications.

References

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

[2] J. Fridrich, M. Goljan, and R. Du, "Invertible authentication watermark for JPEG images," in *Proc. Inf. Technol. Coding Comput.*, Las Vegas, NV, USA, Apr. 2001, pp. 223–227.

[3] H. J. Kim, V. Sachnev, Y. Q. Shi, J. Nam, and H. G. Choo, "A novel difference expansion transform for reversible data embedding," *IEEE Trans. Inf. Forensics Security*, vol. 3, no. 3, pp. 456–465, Sep. 2008.

[4] D. Coltuc, "Improved embedding for prediction based reversible watermarking," *IEEE Trans. Inf. Forensics Security*, vol. 6, no. 3, pp. 873–882, Sep. 2011.

[5] X. Li, B. Ying, 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.

[6] Y. Hu, H. K. Lee, K. Chen, and J. Li, "Difference expansion based reversible data hiding using two embedding directions," *IEEE Trans. Multimedia*, vol. 10, no. 8, pp. 1500–1511, Dec. 2008.

[7] B. Ou, X. Li, Y. Zhao, R. Ni, and Y.-Q. Shi, "Pairwise prediction-error expansion for efficient reversible data hiding," *IEEE Trans. Image Process.*, vol. 22, no. 12, pp. 5010–5021, Dec. 2013.

[8] W. L. Tai, C. M. Yeh, and C. C. Chang, "Reversible data hiding based on histogram modification of pixel differences," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 6, pp. 906–910, Jun. 2009.

[9] C. C. Lin, W. L. Tai, and C. C. Chang, "Multilevel reversible data hiding based on histogram modification of difference images," *Pattern Recognit.*, vol. 41, no. 12, pp. 3582–3591, Dec. 2008.

[10] 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, Jun. 2009.

[11] S. L. Lin, C. F. Huang, M. H. Liou, and C. Y. Chen, "Improving histogram-based reversible information hiding by an optimal weight-based prediction scheme," *J. Inf. Hiding Multimedia Signal Process.*, vol. 4, no. 1, pp. 19–33, Jan. 2013.

[12] S. W. Weng, Y. Zhao, R. R. Ni, and J. S. Pan, "Parity-invariability-based reversible watermarking," *Electron. Lett.*, vol. 45, no. 20, pp. 1022–1023, Sep. 2009.

[13] S. Weng, Y. Zhao, J. S. Pan, and R. Ni, "Reversible watermarking based on invariability and adjustment on pixel pairs," *IEEE Signal Process. Lett.*, vol. 15, no. 20, pp. 721–724, Dec. 2008.

[14] C. F. Lee and Y. L. Huang, "Reversible data hiding scheme based on dual stegano-images using orientation combinations," *J. Telecommun. Syst.*, vol. 52, no. 4, pp. 2237–2247, 2013.

[15] G. Horng, Y. H. Huang, C. C. Chang, and Y. Liu, "(k, n)-image reversible data hiding," *J. Inf. Hiding Multimedia Signal Process.*, vol. 5, no. 2, pp. 152–164, Apr. 2014.

[16] Y. Wang and K. N. Plataniotis, "An analysis of random projection for changeable and privacy-preserving biometric verification," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 40, no. 5, pp. 1280–1293, Oct. 2010.

[17] A. Dabrowski, E. R. Weippl, and I. Echizen, "Framework based on privacy policy hiding for preventing unauthorized face image processing," in *Proc. IEEE Int. Conf. Syst. Man Cybern. (SMC)*, Manchester, U.K., Oct. 2013, pp. 455–461.

[18] Y. T. Wu and F. Y. Shih, "Genetic algorithm based methodology for breaking the steganalytic systems," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 1, pp. 24–31, Feb. 2006.

[19] X. Gao, C. Deng, X. Li, and D. Tao, "Geometric distortion insensitive image watermarking in affine covariant regions," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 40, no. 3, pp. 278–286, May 2010.

[20] M. S. Hsieh and D. C. Tseng, "Image subband coding using fuzzy inference and adaptive quantization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 33, no. 3, pp. 509–513, Jun. 2003.

[21] S. Lian, Z. Liu, Z. Ren, and H. Wang, "Commutative encryption and watermarking in video compression," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 6, pp. 774–778, Jun. 2007.

[22] M. Cancellaro *et al.*, "A commutative digital image watermarking and encryption method in the tree structured Haar transform domain," *Signal Process. Image Commun.*, vol. 26, no. 1, pp. 1–12, Jan. 2011.

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

[24] W. Hong, T. S. Chen, and H. Wu, "An improved reversible data hiding in encrypted images using side match," *IEEE Signal Process. Lett.*, vol. 19, no. 4, pp. 199–202, Apr. 2012.

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

[26] 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.

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

[28] W. Zhang, K. Ma, and N. Yu, "Reversibility improved data hiding in encrypted images," *Signal Process.*, vol. 94, pp. 118–127, Jan. 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. Li, D. Zhang, Z. Liu, and X. Qiao, "Fast block-based image restoration employing the improved best neighborhood matching approach," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 35, no. 4, pp. 546–555, Jul. 2005.
[31] M. Johnson, P. Ishwar, V. Prabhakaran, D. Schonberg, and K. Ramchandran, "On compressing encrypted data," *IEEE Trans. Signal Process.*, vol. 52, no. 10, pp. 2992–3006, Oct. 2004.
[32] W. Liu, W. Zeng, L. Dong, and Q. Yao, "Efficient compression of encrypted grayscale images," *IEEE Trans. Image Process.*, vol. 19, no. 4, pp. 1097–1102, Apr. 2010.
[33] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.
[34] R. Rubinstein, T. Peleg, and M. Elad, "Analysis K-SVD: A dictionary-learning algorithm for the analysis sparse model," *IEEE Trans. Signal Process.*, vol. 61, no. 3, pp. 661–677, Feb. 2013.
[35] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.
[36] G. Sullivan and T. Wiegand, "Video compression—From concepts to the H.264/AVC standard," *Proc. IEEE*, vol. 93, no. 1, pp. 18–31, Jan. 2005.
[37] 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.
[38] W. Stallings, *Cryptography and Network Security: Principles and Practice*, 3rd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2003.
[39] P. Bas, T. Filler, and T. Pevny, "'Break our steganographic system': The ins and outs of organizing BOSS," in *Proc. 13th Int. Conf. Inf. Hiding Conf.*, Prague, Czech Republic, May 2011, pp. 59–70.
[40] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Jun. 2010.
[41] H. Jegou, M. Douze, and C. Schmid, "Hamming embedding and weak geometric consistency for large scale image search," in *Proc. 10th Eur. Conf. Comput. Vis. I (ECCV)*, Marseille, France, Oct. 2008, pp. 304–317.
[42] O. Bryt and M. Elad, "Compression of facial images using the K-SVD algorithm," *J. Vis. Commun. Image Represent.*, vol. 19, no. 4, pp. 270–282, May 2008.

Xiaochun Cao (SM'14) received the B.E. and M.E. degrees from Beihang University (BUAA), Beijing, China, and the Ph.D. degree from the University of Central Florida, Orlando, FL, USA, all in computer science.

He is a Professor with the Institute of Information Engineering, Chinese Academy of Sciences, Beijing. He was a Research Scientist with ObjectVideo Inc., Reston, VA, USA, for about three years. From 2008 to 2012, he was a Professor with Tianjin University, Tianjin, USA. He has authored and co-authored over 80 journal and conference papers.

Prof. Cao was a recipient of the University Level Outstanding Dissertation Award nomination for his dissertation and the Piero Zamperoni Best Student Paper Award at the International Conference on Pattern Recognition in 2004 and 2010.

Ling Du received the B.E. and M.E. degrees in computer science from Liaoning University, Jinzhou, China. She is pursuing the Ph.D. degree with the School of Computer Science and Technology, Tianjin University, Tianjin, China.

She was a Teacher with Shenyang Aerospace University, Shenyang, China, from 2007 to 2011. Her current research interests include multimedia information security and computer vision.

Xingxing Wei received the B.S. degree from the School of Automation and Electrical Engineering, Beihang University, Beijing, China, in 2010. He is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Tianjin University, Tianjin, China, under the guidance of Prof. X. Cao.

His current research interests include structured learning and prediction, tensor analysis and its applications, image retrieval, saliency detection, and object recognition.

Dan Meng (M'02) received the Ph.D. degree in computer science from the Harbin Institute of Technology, Harbin, China, in 1995.

He is currently a Professor with the Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China. His current research interests include high-performance computing and computer architecture.

Prof. Meng is a Senior Member of China Computer Federation.

Xiaojie Guo (M'13) received the B.E. degree in software engineering from the School of Computer Science and Technology, Wuhan University of Technology, Wuhan, China, in 2008, and the M.S. and Ph.D. degrees in computer science from the School of Computer Science and Technology, Tianjin University, Tianjin, China, in 2010 and 2013, respectively.

He is currently an Assistant Professor with the Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China.

Dr. Guo was a recipient of the Piero Zamperoni Best Student Paper Award with the International Conference on Pattern Recognition (International Association on Pattern Recognition) in 2010.