Reversible Data Hiding in Encrypted Binary Images

Amrutha K A[1], Sanish V S[2]

[1]M. Tech. Scholar, [2] Assistant Professor, Electronics and Communication Engineering

Jawaharlal College of Engineering and Technology, Palakkad, Kerala, India

Abstract - A novel reversible data hiding algorithm for encrypted images is proposed. In encryption phase, a binary location sequence is applied to encrypt the original image. Then the least significant bits (LSBs) of pixels in encrypted image are losslessly compressed to leave place for secret data. At receiving terminal, the operation is flexible, that is, it meets the requirement of separation. With the decryption key, a receiver can get access to the marked decrypted image which is similar to the original one. With data-hiding key, the receiver can successfully extract secret data from the marked encrypted image. With both keys, the receiver can get secret data and the exactly original image.

Keywords: Reversible data hiding, Encryption, Binary location sequence.

I. INTRODUCTION

Reversible data hiding (RDH) in images is a technique, by which the original image can be losslessly recovered after the extraction of the secret data. This important technique can be applied to many scenarios, such as law forensics, military imagery and medical imagery, where no distortion of the original image is allowed. For this reason, RDH has attracted considerable research interest. A strategy for RDH is based on lossless compression, in which a data hider makes use of redundancy of the original cover to create a blank space for embedding the secret data. With the increasing demand of privacy protection, encryption becomes an effective and popular means, which converts original image into unintelligible one. There are some applications while RDH can be applied for encrypted images. For instance, to protect the privacy of the patient, the medical images should have been encrypted, a database administrator may aim to embed the personal data into encrypted medical images. With the personal data, the database administrator can manage the images without knowing the original content. On the other hand, a doctor, having both the data-hiding and the encryption keys, can extract the personal data and recover the original content without any error. That means a RDH scheme for encrypted image is attractive.

In some special scenarios, e.g., patent certificate, scanned check, handwritings, and CAD graphs, binary image becomes mainstream formats. Compared with 8 bits per gray pixel or 24 bits per color pixel, binary images always require only 1 bit per pixel (either 0 or 1) so that the embedding process in binary images may lead to more significant distortion on both statistics and vision. Therefore, the RDH issue aiming at encrypted binary images has received considerable critical attention.

We propose a new reversible data hiding solution for encrypted binary images, which can implement data embedding and data extraction with high embedding capacity. We claim that our proposed method can work over natural binary images and can achieve separable operations of data extraction, direct image decryption, and image recovery according to the availability of encryption key and data hiding key. We divide binary image into pure color blocks and non-pure color blocks, which are marked by a binary location sequence. A new halving compression mechanism is introduced to further compress the location sequence. This significantly reduces the length of auxiliary data. Accordingly, the embedding capacity of additional data is further improved. We design a cross-segmentation mechanism that is used to divide the non-pure color block into two-pixel sets: embeddable pixels and prediction pixels. The former is used to embed additional data, while the latter is used to predict and recover the embeddable pixels that have changed during the embedding operation. Since a prediction pixel can be shared with multiple embeddable pixels' recovery, it naturally improves the embedding capacity, while ensuring the correct recovery of embeddable pixels. We perform comprehensive experiments with different binary image sources. The experimental results demonstrate that our solution significantly outperforms existing reversible data hiding methods in terms of embedding capacity and visual quality.

II. SYSTEM MODEL

The proposed RDH-EB framework is mainly comprised of three parts. In the first part, we first divide binary image into pure color blocks and non-pure color blocks, which are rearranged by placing pure color blocks in the back and non-pure color blocks in the front. A binary location sequence which marked these image blocks is further

lossless compressed as auxiliary data by introducing a new halving compression mechanism. In addition, each non-pure color block is divided into two-pixel sets, embeddable pixels and prediction pixels, by a cross-segmentation mechanism. The former is used to embed additional data, while the latter is used to predict and recover the embeddable pixels that have changed during embedding operation. Finally, the rearranged binary image is encrypted and then is delivered to the data hider. In the second part, additional data are embedded into the encrypted binary image, which is transmitted to the receiver over network. In the third part, according to the availability of encryption key and data hiding key, the receiver can achieve separable operations of data extraction, direct image decryption, and image recovery. The framework is shown in Fig

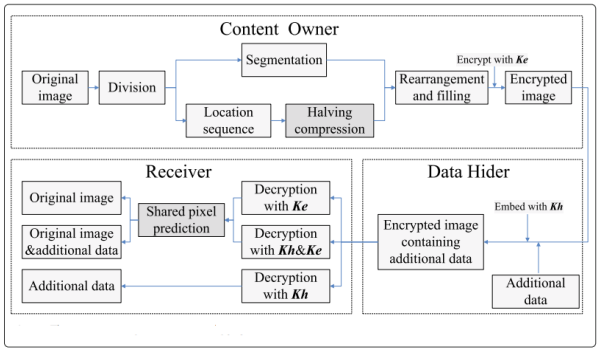

Fig no:01 Framework of proposed model

III. PREVIOUS WORK

Existing binary image-based RDH solutions are mainly divided into two categories: plaintext binary image-based RDH schemes and encrypted binary image-based RDH schemes. We briefly introduce existing works according to two criteria. Firstly, regarding plaintext binary image-based RDH schemes, Wang et al. [19] proposed a large volume data hiding scheme of binary image based on block pattern. In this scheme, Wang et al. divide cover binary image into non-symmetrical block and dual-pair block, and then design two matching pairs (MP), internal and external, to reduce embedded changes. Finally, the internal adjustment matching pair (MPIA) method is performed sequentially on the embeddable blocks, which are selected by a randomly generated seed and output a marked image. Although the embedding capacity is limited, Wang's scheme can still achieve better performance with the same embedding distortion.

Secondly, regarding encrypted binary image-based RDH schemes, in [20], Ren et al. proposed a new reversible data hiding method in encrypted binary images by pixel prediction. The original binary image is divided into non-overlapping uniform blocks and non-uniform blocks, which are effectively marked by an auxiliary type image including 0 and 1. The type image is embedded by content owner, and secret data is embedded in these blocks by data hider. For the receiver, after extracting the secret data by data embedding key, the original image can be recovered by a "T" pattern and the type image. Ren's method works well, but the visual quality and embedding rate can be further improved. This is mainly because the "T" pattern involves more different neighbour pixels to predict one embeddable pixel, resulting in an obvious waste of embedding space. In addition, since the auxiliary information is directly embedded into image without any compression, the embedding capacity is further reduced.

Accordingly, a series of works have been developed to address the problem of applying reversible data hiding in binary images. Since these schemes do not make full use of the prediction performance of neighbouring pixels, and also do not consider to optimize the compression space of auxiliary data, they thus have a weak visual quality and lower embedding capacity. This paper tries to further fill this gap

IV. PROPOSED METHODOLOGY

The proposed RDH-EB framework is mainly comprised of three parts. In the first part, we first divide binary image into pure color blocks and non-pure color blocks, a binary location sequence which marked these image blocks is further lossless compressed. The rearranged binary image is encrypted and then is delivered to the data hider. In the second part, additional data are embedded into the encrypted binary image, which is transmitted to the receiver over network. In the third part, according to the availability of encryption key and data hiding key, the receiver can achieve separable operations of data extraction, direct image decryption, and image recovery

IMAGE PRE-PROCESSING

Given an original binary image with size M × M, we firstly divide this image into multiple non-overlapping blocks with size N × N, N < M. Since the pixels of binary image are either black or white, we denote the N × N blocks composed of all black or all white pixels as pure color blocks (PB for short). Similarly, we denote other N ×N blocks as nonpure blocks (NPB for short). Obviously, if any one pixel in PB block is confirmed, we can fastly determine the color information of all other pixels in this block. It implies that we can use only one pixel to record the color information of PB, while the remaining pixels can be considered as embedding room. Subsequently, we introduce a cross-segmentation mechanism to preprocess NPB blocks. For each NPB, all pixels are divided across into two sets: embeddable pixels and shared prediction pixels. The former is used to embed secret data, while the latter is used to predict the original embeddable pixels if they are modified due to data hiding. In our scheme, since

the shared prediction pixels are distributed across in each NPB, one shared prediction pixel can be used to predict multiple embeddable pixels, thereby reserving more embeddable pixels to carry secret data. Figure 2 presents the embeddable pixels and the shared prediction pixels, which are represented by plus symbol and dot symbol, respectively.

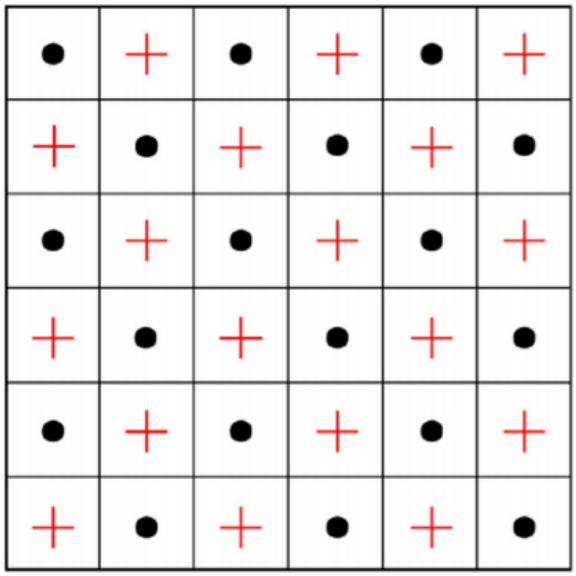

Fig.02: Dot set and plus set in a non-pure block. The plus set stands for embeddable pixels, while dot set represents the shared prediction pixels

Subsequently, we denote PB block as 1 and NPB block as 0 and then scan the original image from left to right, top to bottom to generate a series of binary bits, which is denoted as location sequence. Clearly, the location sequence has a fixed length M2 N2 according to the size of original image. For easy understanding, an example is shown in Fig. 03.

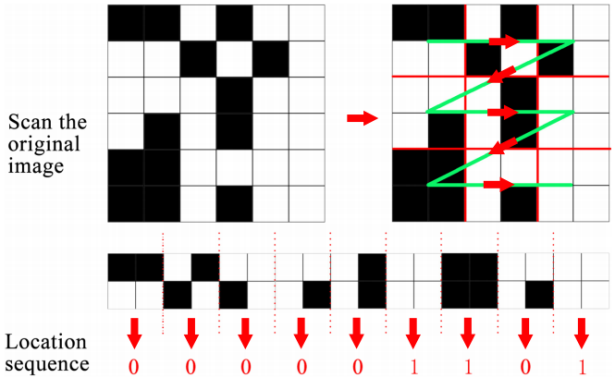

Fig.03. Binary location sequence generation. With this location sequence, the position of each block can be located easily in the original image

AUXILIARY DATA COMPRESSION

The generated binary location sequence indicates the position of each block in original image; we denote it as auxiliary data and deliver it to the receiver to achieve the original image recovery. Apparently, if the binary location sequence is too long, it inevitably occupies the embedding capacity of additional data. In order to avoid this problem, we design a lossless compression method, named as halving compression, to further reduce the length of binary location sequence.

First, we set two thresholds T ∈ {T0, T1} to perform the compression of consecutive 0 and 1, respectively. A quotient list and a remainder list are used to save the compressed data. Denote the number of consecutive 1 (or 0) in original location sequence as Lt, the compression procedure can be shown as follows in Eq. (1):

$$Q_t = \begin{cases} L_t, & \text{if } L_t < T \\ \lfloor \frac{L_t+T}{2} \rfloor, & \text{otherwise} \end{cases} \quad (1)$$

$$R_t = \mathrm{mod}\left((L_t + T), 2\right), \quad T \in \{T_0, T_1\} \quad (2)$$

In this equation, when T is set to T1, Lt indicates the number of consecutive 1 in the original location sequence and Qt stands for the number of compressed 1, which is added directly to the quotient list. Similarly, when T is set to T0, Lt and Qt correspond to the case of consecutive 0. In addition, if Lt ≥ T, we calculate Rt as Eq. (2) to represent the remainder of 1 or 0 and add it to the remainder list. Finally, the compressed location sequence can be directly obtained by splicing remainder list and quotient list. The process can be divided into five steps:

- Step 1: The same continuous bits “xxxxx” are scanned.
- Step 2: Use Eq. (1) to calculate the quotient and then add it to the quotient list.
- Step 3: Use Eq. (2) to calculate remainder and then add it to the remainder list.
- Step 4: Repeat steps 1–3 until all bits are processed.
- Step 5: Finally, the remainder list is spliced after the quotient list to form a compressed bit stream.

In order to make it easier to understand, an actual example is shown in Fig. 4. As shown in Fig. 4, the original bitstream is “110100000111”, T1 = 2, T0 = 2. And the specific operation of this example is as follows:

- Step 1: The same continuous bits “11” are scanned.
- Step 2: Use Eq. (1) to calculate quotient as 2. Then, add 2 bits “1” to the quotient list.
- Step 3: Use Eq. (2) to calculate remainder as 0. Then, add the result to the remainder list.

- Step 4: Repeat steps 1–3 until all bits are processed. At this step, the quotient list is 11010001111, and the remainder list is 010.
- Step 5: Finally, the remainder list is spliced after the quotient list to form a compressed bit stream

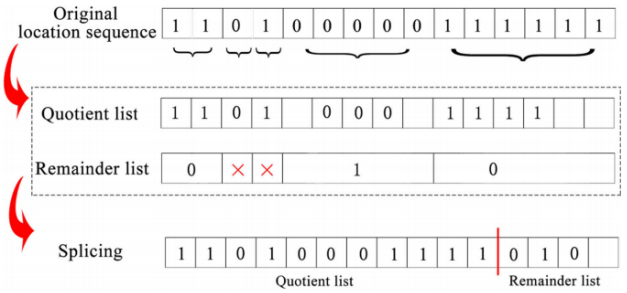

Fig.04. An example for proposed halving compression algorithm with T1 = 2, T0 = 2. The original location sequence 110100000111111, quotient list 11010001111, and remainder list 010. The compressed location sequence 11010001111010 can be directly obtained by combining remainder list and quotient list

The decompression procedure can be shown as the following four steps.

$$L_t = \begin{cases} Q_t, & \text{if } Q_t < T \\ Q_t \times 2 + R_t \times b, & \text{otherwise} \end{cases} \quad (3)$$

$$b = \begin{cases} 1, & \text{if } Q_t = \text{“1...”} \\ 0, & \text{if } Q_t = \text{“0...”} \end{cases} \quad (4)$$

- Step 1: The continuous compressed bit sequence “xxxxx” is scanned from quotient list.
- Step 2: Based on the above compressed bit sequence, use Eqs. (3) and (4) to calculate the corresponding results as a part of original location sequence Lt.
- Step 3: Repeat steps 1 and 2 until all bits in the quotient list are processed completely.
- Step 4: Finally, sequential merging results of steps 1–3 to form the original location sequence.

IMAGE ENCRYPTION

Before performing image encryption, we sequentially implement three measures to further improve security performance. First, we rearrange these blocks by placing the NPBs in the front and the PBs in the back. Second, the compressed location sequence is embedded sequentially from the first PB block. Third, we record a fixed pixel in each PB block (e.g., the pixel in bottom right corner) as the recovering pixel and keep it unchanged. Then, the remaining pixels in each PB block, together with the pixels of plus set in NPBs, are filled by random bits. This operation can efficiently resist known plaintext attacks. Finally, we denote the processed image as a rearranged image. Subsequently, we use stream cipher to encrypt the rearranged image. Denote x = {x1x2 ⋯ xk ⋯ xM×M} as a vector that represents all pixel values of rearranged image, 1 ≤ k ≤ M × M. Generating a binary bit sequence r = {r1r2 ⋯rk ⋯rM×M} with the encryption key Ke, the pixel is then encrypted as xˆk.

$$\hat{x}_k = x_k \oplus r_k, \qquad 1 \le k \le M \times M \quad (5)$$

Since the stream cipher only performs XOR processing, the position of each pixel in the image is not changed after encryption.

DATA EMBEDDING

In this section, we describe the procedure of data embedding. When data hider receives the encrypted image, he can use the data hiding key Kh to randomly select some of the following two types of pixels to embed additional data

- For PB blocks, apart from the recovering pixels (e.g., the pixel marked by the symbol in Fig 5a) and the pixels carrying the auxiliary data, other pixels can be directly replaced by additional In data.
- For NPB blocks, the embeddable pixels (which correspond to the plus set in Fig. 2) can be directly replaced by additional data, while the shared prediction pixels remain unchanged. The embedding positions can refer to Fig. 5b.

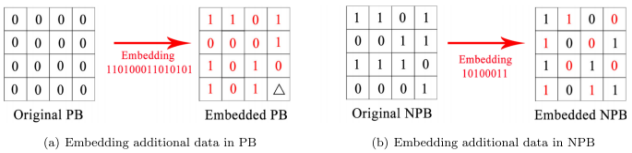

Fig.05. Embedding additional data in 4 × 4 PB block and NPB block. The symbol indicates the recovery pixel, which is not used to embed data

Notably, according to the embedding algorithm, the maximum embedding capacity of PB blocks, denoted by Mp, can be calculated as follows:

$$M_p = (N^2 - 1) \times N_p - L_c \quad (6)$$

where N is the size of block, Np represents the number of PB blocks, and Lc stands for the length of compressed location sequence. Regarding NPB blocks, since only half of the pixels in each block are used to embed data, its maximum embedding capacity Mn can be also calculated easily:

$$M_n = \frac{N^2}{2} \times N_n \quad (7)$$

where Nn represents the number of NPB blocks. Finally, the total maximum embedding capacity (MEC for short) can be calculated by combining Eqs. (6) and (7). Accordingly, the maximum embedding rate (MER for short) can be expressed as follows:

$$N_n + N_p = \frac{M^2}{N^2} \tag{8}$$

$$\begin{aligned} \mathrm{MER} &= \frac{M_p + M_n - L_c}{M^2} \\ &= \frac{\left(\frac{N^2}{2} - 1\right) \times N_p - L_c}{M^2} + \frac{1}{2} \end{aligned} \tag{9}$$

Obviously, the block size N and compression thresholds T1 and T0 in the above description have a significant influence on embedding capacity. They will be discussed in the experimental section.

DATA EXTRACTION AND IMAGE RECOVERY

In this section, we discuss the data extraction and image recovery procedure. When the encrypted image containing the secret messages is received by the recipient, he can separably achieve three operations, data extraction, direct image decryption, and image recovery, according to the cases of obtaining the key

Case 1: Only obtaining the data hiding key

If the recipient only obtained the data hiding key, according to the algorithm, the additional data can be directly extracted, even if the image has not been decrypted.

Case 2: Only obtaining the encryption key

If the recipient only obtained the encryption key, due to the XOR process in encryption procedure, he can perform decryption procedure directly by the encryption key and extract the compressed location sequence from the decrypted image. Accordingly, the original image can be also recovered by the following steps:

- Step 1: Decrypt the received image and then decompress the extracted location sequence. With the location sequence, the positions of each block can be arranged correctly.
- Step 2: For each PB block, it can be directly recovered based on the fixed recovery pixel For each NPB block, since the shared prediction pixels have no changes during data embedding, the embeddable pixel P(i,j) can be calculated by its three adjacent shared prediction pixels D(i,j−1), D(i,j+1), and D(i+,j).

$$P_{(i,j)} = \begin{cases} 0, & \text{if } \frac{D_{(i,j-1)} + D_{(i,j+1)} + D_{(i+1,j)}}{3} < 0.5 \\ 1, & \text{otherwise} \end{cases} \tag{10}$$

- Step 3: Repeat step 2 until all image blocks are recovered. The recipient can finally obtain the original binary image.

V. EXPERIMENTAL RESULTS

A series of experiments is carried out on classical binary image sources, such as artoon, CAD, Texture, Mask, Handwriting and Document. These images are sequentially converted to binary images. All experimental images are resized to 256 × 256. The performance of the proposed scheme is tested in terms of visual quality, The peak signal to noise ratio (PSNR) and structural similarity (SSIM) are used to measure the visual quality. In specific, denote the original

image as **X** and the decrypted image as **I**:

$$\mathrm{MSE} = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} ||\mathbf{X}(i,j) - \mathbf{I}(i,j)||^2 \tag{11}$$

$$\mathrm{PSNR} = 10 \cdot \log_{10} \left(\frac{\mathrm{MAX}_\mathbf{I}^2}{\mathrm{MSE}} \right) \mathrm{dB} \tag{12}$$

where MSE is the mean square error between **X** and **I**. MAX**I** = 2 represents the maximum value of color due to 2-bit representation in binary image. In general, the larger the PSNR value, the less the distortion and the higher the visual quality. In addition, the structural similarity is also considered as another standard measurement, which can be calculated as follows:

$$\mathrm{SSIM}(\mathbf{X}, \mathbf{I}) = \frac{(2\mu_\mathbf{X}\mu_\mathbf{I} + c_1)(2\sigma_{\mathbf{XI}} + c_2)}{(\mu_\mathbf{X}^2 \mu_\mathbf{I}^2 + c_1)(\sigma_\mathbf{X}^2 + \sigma_\mathbf{Y}^2 + c_2)} \tag{13}$$

where μ**X** and μ**I** are the average values of original image and the decrypted image, σ**X** and σ**I** are the standard deviation values of two images, and σ**XI** represents the covariance

of the two images. $c1$ and $c2$ are the constants for stable division. In general, SSIM value is between 0 and 1. If SSIM value is close to 1, the decrypted image is like the original image with high quality. Moreover, we define embedding rate (ER) as bpp (bits per pixel), that is, each pixel can carry the average number of bits.

Test for visual quality

In this section, we test the visual quality performance of the proposed scheme. For easy illustration, we select Cartoon to validate the effectiveness of proposed method, where Cartoon is from classical binary image set, The experimental results are shown in Fig. 6.

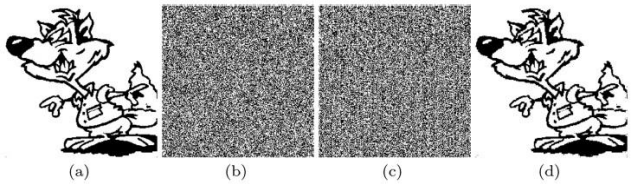

Fig 6 (a) Original Image (b) Encrypted Version (c) Encrypted image containing additional Data (d) Recovered Image

As can be seen from this figure, the proposed method can extract the secret data correctly and obtain the recovery image that is very similar to the original image. This demonstrates that our method can effectively achieve the reversibility, that is, extract secret data correctly while ensuring that the original image is perfectly recovered.

Furthermore, we test PSNR and SSIM values of recovery image with *N* changing. In this test, each image is implemented data embedding with its maximum embedding rate. The testing results are shown in Fig. 7. From this figure, it is clear that the highest PSNR and SSIM can be obtained when the block size is the minimum ($N = 2$). This is because the minimum block size may produce the most unavailable pixels that cannot be used to embed additional data (including the recovering pixels in PB blocks and the shared prediction pixels in NPB blocks), finally resulting in highest visual quality of recovery image.

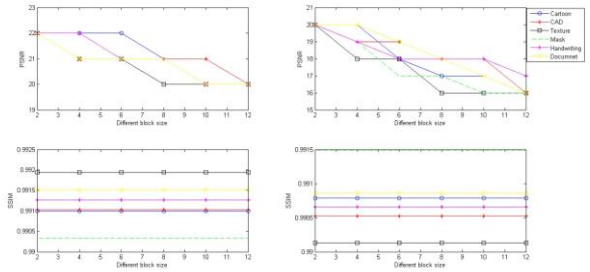

Fig 7 Comparison of the PSNR and SSIM values with different block size N

VI. CONCLUSION

In this paper, we addressed the reversible data hiding problem in encrypted binary image and proposed a new scheme with shared pixel prediction method and halving compression mechanism, which has a significant superior performance. We firstly designed a shared pixel prediction method by introducing a cross-segmentation mechanism on non-pure color blocks. Since a shared pixel can be used to recover multiple embeddable pixels, the embedding capacity is naturally improved comparing with Ren's scheme. Furthermore, a lossless halving compression mechanism is introduced to compress the location sequence, which is used to mark the position of image block and helps to further increase the embedding capacity. Experimental results show that both the shared pixel prediction method and halving compression mechanism contribute to the improvement of embedding capacity. Compared to the existing RDH solutions in binary images, our method has superior performance when the embedding capacity is large.

VII. FUTURE SCOPES

In the future, we plan to carry our work forward in two directions. First, compared to the same block size in the proposed scheme, we try to design an adaptive segmentation mechanism for the pure color blocks. This may further add the embedding capacity due to involving less recovering pixels.

REFERENCES

[1] J. Tian, in *Security and Watermarking ofMultimedia Contents IV, vol. 4675*, Wavelet-based reversible watermarking for authentication International Society for Optics and Photonics, (2002), pp. 679–690

[2] J. Tian, Reversible data embedding using a difference expansion. IEEE Trans. Circ. Syst. Video Technol. **13**(8), 890–896 (2003)

[3] D. M. Thodi, J. J. Rodríguez, Expansion embedding techniques for reversible watermarking. IEEE Trans. Image Process. **16**(3), 721–730 (2007)

[4] L. Kamstra, H. J. Heijmans, Reversible data embedding into images using wavelet techniques and sorting. IEEE Trans. Image Process. **14**(12), 2082–2090 (2005)

[5] Z. Ni, Y.-Q. Shi, N. Ansari, W. Su, Reversible data hiding. IEEE Trans. Circ. Syst. Video Technol. **16**(3), 354–362 (2006)

[6] C. Qin, W. Zhang, F. Cao, X. Zhang, C.-C. Chang, Separable reversible data hiding in encrypted images via adaptive embedding strategy with block selection. Signal Process. **53**, 109–122 (2018)

[7] Z. Ni, Y. Q. Shi, N. Ansari,W. Su, Q. Sun, X. Lin, in *2004 IEEE International Conference on Multimedia and Expo (ICME)(IEEE Cat. No. 04TH8763)*, Robust lossless image data hiding, vol. 3, (2004), pp. 2199–2202

[8] Z. Ni, Y. Q. Shi, N. Ansari, W. Su, Q. Sun, X. Lin, Robust lossless image data hiding designed for semi-fragile image authentication. IEEE Trans. Circ. Syst. Video Technol. **18**(4), 497–509 (2008)

[9] Z. Qian, X. Zhang, S. Wang, Reversible data hiding in encrypted jpeg bitstream. IEEE Trans. Multimed. **16**(5), 1486–1491 (2014)

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

[11] H. Wu, F. Li, C. Qin, W. Wei, Separable reversible data hiding in encrypted images based on scalable blocks. Multimed. Tools Appl. **78**(18), 25349–25372 (2019)

[12] Y. Qiu, Z. Qian, H. Zeng, X. Lin, X. Zhang, Reversible data hiding in encrypted images using adaptive reversible integer transformation. Signal Process. **167**, 107288 (2020)

[13] W. Lu, L. He, Y. Yeung, Y. Xue, H. Liu, B. Feng, Secure binary image steganography based on fused distortion measurement. IEEE Trans. Circ. Syst. Video Technol. **29**(6), 1608–1618 (2018)

[14] K. Ma, W. Zhang, X. Zhao, N. Yu, F. Li, Reversible data hiding in encrypted images by reserving room before encryption. IEEE Trans. Inf. Forensics Secur. **8**(3), 553–562 (2013)

[15] W. Zhang, K. Ma, N. Yu, Reversibility improved data hiding in encrypted images. Signal Process. **94**, 118–127 (2014)

[16] C. Shiu, Y. Chen, W. Hong, Encrypted image-based reversible data hiding with public key cryptography from difference expansion. Signal Process. Image Commun. **39**, 226–233 (2015)

[17] X. Cao, L. Du, X. Wei, D. Meng, X. Guo, High capacity reversible data hiding in encrypted images by patch-level sparse representation. IEEE Trans. Cybern. **46**(5), 1132–1143 (2015)

[18] Z. Ma, F. Li, X. Zhang, Data hiding in halftone images based on hamming code and slave pixels. J. Shanghai Univ. **19**(2), 111–115 (2013)

[19] C.-C. Wang, Y.-F. Chang, C.-C. Chang, J.-K. Jan, C.-C. Lin, A high capacity data hiding scheme for binary images based on block patterns. J. Syst. Softw. **93**, 152–162 (2014)

[20] H. Ren, W. Lu, B. Chen, Reversible data hiding in encrypted binary images by pixel prediction. Signal Process. **165**, 268–277 (2019)