AN IMPROVED REVERSIBLE DATA HIDING IN ENCRYPTED IMAGES

*Shuang Yi, Yicong Zhou**

Department of Computer and Information Science
University of Macau, Macau 999078, China

ABSTRACT

This paper is an improved version of Zhang's reversibility improved data hiding method in encrypted images. The original work randomly selects $p\%(0 < p \leqslant 20)$ pixels from an original image to obtain the estimation error for secret data embedding. In this work, we estimate half of the pixels in the original image to obtain the estimation error so that the maximum embedding rate can be significantly improved while keeping a high image quality of the marked decrypted image. The experimental results show that our proposed method has a higher performance than Zhang's method.

***Index Terms*—** reversible data hiding, prediction error, histogram shifting, image encryption

1. INTRODUCTION

Recently, embedding secret data in encrypted images has caught many people's attention. It enables the data hider to embed secret data into an encrypted image without knowing the original image content, where the encrypted image is generated by the content owner. The receiver with different security keys can obtain different contents (secret data, original image or both). This is very useful in many applications such as cloud storage and medical image management. The content owner encrypts the original image for privacy protection, for the cloud provider or administration, he may hope to append some data into the image for labeling or classification, and he has no right to access the original image. Only the authorized users can obtain the decrypted image and/or appended data with different security keys.

Existing reversible data hiding methods in encrypted images (RDHEI) can be divided into two categories: without or with a preprocessing before image encryption. For the former one, the content owner does nothing except for encrypting the image. Puech *et al.* [1] proposed an RDHEI method which embeds one secret data bit in an encrypted image block by bit substitution, image recovering is based on analyzing the local standard deviation of the decrypted image. This method has low embedding rate. In [2], original image is first encrypted by a bit XOR, then one bit of the secret data is embedded into one image block by flipping the 3 least significant bits (LSBs), data extraction/image recovering is accomplished by exploiting the pixel correlations. This method may suffer incorrect data extraction and image recovering results when image block is small (e.g., 8×8). To reduce the incorrect rate of data extraction and image recovering, Hong *et al* [3] proposed an improved version by modifying the block smoothness estimation function. In [4], Zhang proposed an RDHEI method that data extraction and image recovering can be performed separately. It empties out spare space for secret data embedding by compressing several LSB planes, and recovers the image by exploiting the spatial correlation in a nature image. In [5], Zhang *et al.* proposed an RDHEI method by compressing half of the pixels in the 4^{th} LSB plane of the encrypted image using the low-dentist parity-check (LDPC) code. Image recovering is accomplished by exploiting the side information provided by the unchanged data. In Qian *et al*'s method in [6], the original image is encrypted by pixel scrambling and mapping, secret data bits are first turned into an n-nary array and then embedded into the encrypted image by histogram shifting method. In [7], Wu *et al* proposed a joint method and a separable method. For the joint method, image is encrypted by a stream cipher, one secret bit is embedded in a group of pixels in encrypted image by modifying the $i^{th}(1 \leqslant t \leqslant 6)$ LSBs. Data extraction and image recovering are accomplished by comparing the estimation error. For the separable method, one secret bit is embedded into one pixel of encrypted image by replacing the $i^{th}(7 \leqslant t \leqslant 8)$ LSBs. Image recovering is accomplished by comparing the estimation error or filtering the directly decrypted image. This method may not extract error-free secret data and recovered image if the original image contains more textures. In order to improve the capacity and recovered image quality, Ma *et al.* [8] introduced another type of RDHEI by emptying out spare room for secret data embedding before image encryption. It embeds several LSBs in a part of the image into the rest of the image using a traditional RDH method. This method can obtain the secret data and recovered image without any error. For Zhang's method in [9], $p\%(0 < p \leqslant 20)$ pixels from

This work was supported in part by the Macau Science and Technology Development Fund under Grant FDCT/017/2012/A1 and by the Research Committee at University of Macau under Grants MYRG2014-00003-FST, MRG017/ZYC/2014/FST, MYRG113(Y1-L3)-FST12-ZYC and MRG001/ZYC/2013/FST.

*Corresponding author. Tel.: (853)8822-8458; Fax: (853)8822-2426, Email address: yicongzhou@umac.mo

the original image are randomly selected and estimated by their surrounding pixels, secret data bits are embedded into the encrypted estimation errors. Two different schemes, data extraction before or after image recovering, are provided to cope with different applications. This method can achieve a high marked decrypted image quality. However, the embedding rate is relatively low. The average embedding rate tested on 50 images is 0.039 when 20% of the pixels are selected for estimation [9].

In this paper, we propose an improved method of [9]. We estimate half of the pixels in original image using the rest pixels that the embedding rate can be significantly improved while keeping a high image quality of the marked decrypted image.

The rest of this paper is organized as follows: Section 2 will introduce the proposed method, Section 3 will show the experimental results and comparisons, Section 4 will draw a conclusion.

2. PROPOSED ALGORITHM

The framework of the proposed algorithm is shown in Fig. 1. The main idea of this method is first to estimate a part of the pixels in an original image using the rest pixels and obtain the estimation errors. Then we encrypt the estimation errors and the rest pixels separately using the encryption key. The data hider then embeds the secret data into the encrypted estimation errors using the data hiding key and scrambles the image using the sharing key. At the receiver side, the secret data and original image can be extracted and recovered separately by using different security keys.

The algorithm is composed of three phases: 1) generation of encrypted image; 2) data hiding; 3) data extraction and image recovering. These three phases are accomplished by the content owner, data hider and receiver, respectively. In the third phase, we provides two cases, data extraction before and after image recovering, to meet different applications. Next, we present our algorithm in detail.

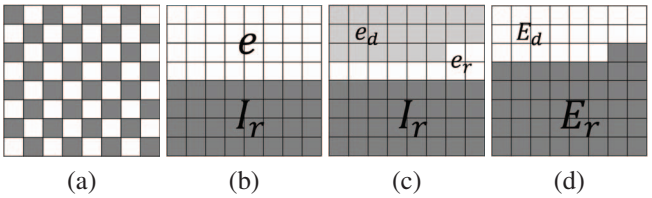

Fig. 2: An illustration of (a) chessboard estimation; (b) estimated results; (c) image before encryption and (d) final version of encrypted image E.

2.1. Generation of Encrypted Image

This phase can be divided into two steps: image estimation and image encryption. We first obtain the estimation errors of half of the pixels in an original image, and then encrypt estimation errors and the remaining pixels of the image separately.

2.1.1. Image Estimation

Assume that an original image $I_{M\times N}$ with a data range of $[0, 255]$ is first divided into two sets like a chessboard: the white set and the black set as shown in Fig. 2(a). We estimate the pixels $I_{(x,y)}$ in white set using the surrounding four pixels in black set by

$$\hat{I}_{(x,y)} = \left\lfloor \frac{I_{(x,y-1)} + I_{(x,y+1)} + I_{(x-1,y)} + I_{(x+1,y)}}{4} \right\rfloor \quad (1)$$

and obtain the estimation error $e_{(x,y)}$ of white set by

$$e_{(x,y)} = \hat{I}_{(x,y)} - I_{(x,y)} \quad (2)$$

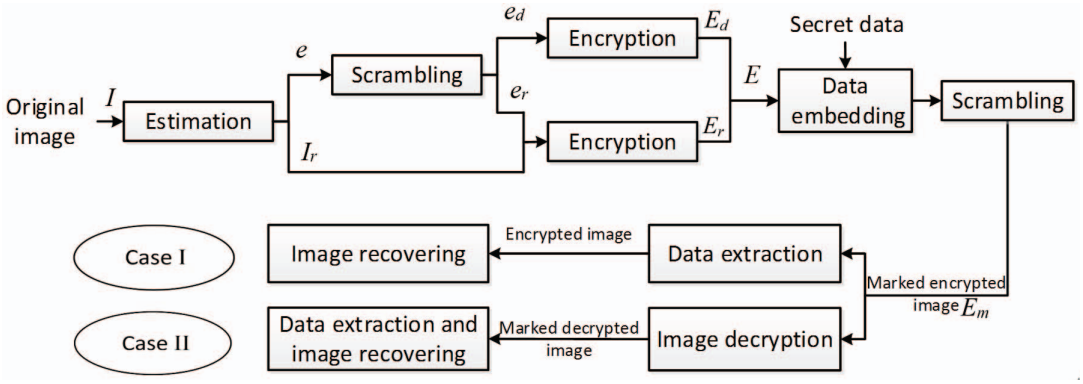

Fig. 1: The framework of proposed the algorithm.

We rearrange $e_{(x,y)}$ to the top of the image and the black set pixels, denoted by $I_{r(x,y)}$, to the bottom of the image as shown in Fig. 2(b).

After obtaining the estimation errors, we scramble them using the encryption key. Then we separate the scrambled $e_s = [e_1, e_2, ..., e_k, e_{k+1}, ..., e_{0.5MN}]$ into two parts denoted as e_d and e_r, respectively, where $e_d = [e_1, e_2, ..., e_k]$ and $e_r = [e_{k+1}, e_{k+2}, ..., e_{0.5MN}]$. Here e_d is utilized to embed secret data and the length of e_d, k, obeys the constraint that

$$e_k \in \{-1, 0\} \quad \text{and} \quad h(-1, e_d) + h(0, e_d) = C \tag{3}$$

where $h(x, z)$, $x \in [-127, 127]$, denotes the histogram value of x in set z and $C = \lceil rMN \rceil$ denotes the capacity of secret data based on a given embedding rate r (.bpp).

2.1.2. Image Encryption

After obtaining e_d and e_r, we encrypt them separately. For e_d, we first shift its values by

$$\hat{e}_i = \begin{cases} e_i + 1 & \text{if } e_i \geqslant 1 \\ e_i - 1 & \text{if } e_i \leqslant -2 \\ e_i & \text{otherwise} \end{cases} \qquad (i = 1, ..., k) \tag{4}$$

to obtain the shifted result $\hat{e}_d$.

Obviously, the estimation error can be either positive or negative value, we use the most significant bit (MSB) to indicate the sign bit, (e.g., 0 for positive and 1 for negative value) thus only the estimation error values falling into $[-127, 127]$ can be successfully stored with 8 bits. For $\hat{e}_d$, we keep the values in the range of $[-124, 125]$ unchanged, truncate the values to $-124(125)$ if it is less(larger) than $-124(125)$, and record the locations and values of truncated errors in a overflow(we use overflow instead of overflow and underflow for the sake of simplicity in this paper) map O_1. All overflow maps generated in this algorithm will be embedded into I_r for reversible recovering at the receiver side. In this paper, we use the RDH method in [10] to embed the overflow map for experiments in next section.

Next, we encrypt $\hat{e}_d$ to obtain the encrypted results $\tilde{e}_d = [\tilde{e}_1, \tilde{e}_2, ..., \tilde{e}_k]$ by

$$\tilde{e}_i = \begin{cases} ((r_r + \hat{e}_i) \mod 126) + 2 & \text{if } \hat{e}_i \geqslant 2 \\ -((r_l - \hat{e}_i) \mod 125) - 3 & \text{if } \hat{e}_i \leqslant -3 \\ \hat{e}_i & \text{otherwise} \end{cases} \tag{5}$$

where $i = 1, 2, ..., k$, $r_r \in [0, 125]$ and $r_l \in [0, 124]$ are pseudo-random sequences generated by the encryption key.

After histogram shifting and encryption, the bins -2 and 1 of $\tilde{e}_d$ are emptied out and we can embed secret data into them by

$$\ddot{e}_i = \begin{cases} 2\tilde{e}_i + m & \text{if } \tilde{e}_i \in \{-1, 0\} \\ \tilde{e}_i & \text{otherwise} \end{cases} \tag{6}$$

where $m \in \{0, 1\}$ denotes the embedded bit and it can be extracted by

$$m = \ddot{e}_i - 2\lfloor \ddot{e}_i / 2 \rfloor \qquad \text{if } \ddot{e}_i \in \{-2, -1, 0, 1\} \tag{7}$$

The error values can be recovered by

$$\tilde{e}_i = \begin{cases} 0 & \text{if } \ddot{e}_i \in \{0, 1\} \\ -1 & \text{if } \ddot{e}_i \in \{-2, -1\} \\ \ddot{e}_i & \text{otherwise} \end{cases} \tag{8}$$

In order to know the length of secret data bits that the data hider can embedded into, we convert the capacity C into a 20-bit sequence and embed it into the first 20 error values of 0 and -1 in $\ddot{e}_i$ using Eqn. (6). The encrypted results $\tilde{e}_d$ with embedded capacity C is denoted as E_d.

For the remaining errors e_r, we first convert its values to $-127(127)$ if it is less(larger) than $-127(127)$ and save the overflow values and locations to another map O_2. Then, we convert all values in e_r to positive values by adding 128 to its absolute values of negative error values. Next, we concatenate e_r and I_r, denoted by as R, and encrypt it by

$$E_r = R \oplus r_2 \tag{9}$$

where $'\oplus'$ is bit-level XOR operation and $r_2 \in [0, 255]$ is a pseudo-random sequence generated by the encryption key.

The final encrypted image E is generated by concatenating E_d and E_r as shown in Fig. 2(d).

2.2. Data Hiding

After obtaining the encrypted image E, the data hider first obtain the side information of capacity C from the first 20 error values of -2, -1, 0 and 1 using Eqn. (7) and recovers these error values using Eqn. (8). Then he encrypts the secret data using the data hiding key and embeds it to the encrypted prediction error E_d that without the information of C using Eqn. (6). After embedding the secret data, the data hider scrambles the image using a sharing key to prevent unauthorized access. Thus the final marked encrypted image is generated and denoted as E_m. The capacity C should be sent to the receiver as a part of the sharing key for data extraction and image recovering.

2.3. Data Extraction and Image Recovering

After obtaining the marked encrypted image E_m, the receiver with different security keys can obtain different contents. Besides, data extraction and image recovering can be accomplished in different orders.

Case 1: *Data extraction before image recovering*

If the receiver only has the data hiding key, he can extract the secret data without knowing the original image content. With the sharing key, he first inverse scrambles E_m, extracts

the encrypted secret data from E_d using Eqn. (7) and decrypts it using the data hiding key to obtain the original secret data. If the receiver holds the encryption key, he can further recover the regional image. He first decrypts E_r using

$$R = E_r \oplus r_2 \tag{10}$$

where r_2 is generated by the same way in image encryption phase. Then he separates R into two parts, I_r and e_r, extracts overflow maps O_1 and O_2 from I_r and recovers e_r using O_2. For the encrypted error values E_d, he first decrypts it using the following equation

$$\hat{e}_i = \begin{cases} (\tilde{e}_i - 2 - r_r) \mod 126 & \text{if } \tilde{e}_i \geqslant 2 \\ -((-\tilde{e}_i - 3 - r_l) \mod 125) & \text{if } \tilde{e}_i \leqslant -3 \\ \tilde{e}_i & \text{otherwise} \end{cases} \tag{11}$$

where $i = 1, 2, ..., k$, r_r and r_l are generated by the same way that mentioned previously. Then the receiver recovers the overflow values using overflow map O_1 and recovers error values $\hat{e}_i$ using Eqn. (8) and inverse shifts them by Eqn. (12) to obtain the recovered error values $e_d = [e_1, e_2, ..., e_k]$.

$$e_i = \begin{cases} \hat{e}_i - 1 & \text{if } e_i \geqslant 2 \\ \hat{e}_i + 1 & \text{if } e_i \leqslant -3 \\ \hat{e}_i & \text{otherwise} \end{cases} \quad (i = 1, ..., k) \tag{12}$$

Then, the receiver concatenates e_r and e_d and inverse scrambles them using the encryption key to obtain the original estimation error e. Finally, he rearranges the recovered error values e and pixels in I_r to their original positions and recovers the pixels in the black set by

$$I_{(x,y)} = \hat{I}_{(x,y)} + e_{(x,y)} \tag{13}$$

where $\hat{I}_{(x,y)}$ is calculated by Eqn. (1). Thus, the recovered image is generated.

Case 2: *Data extraction after image recovering*

In some applications, the original image should be recovered before extracting the secret data. In this scenario, the receiver first inverse scrambles the image and separates it into E_r and E_d based on the sharing key. Then he decrypts E_r using Eqn. (10) and separates the decrypted results into I_r and e_r, extracts overflow maps O_1 and O_2 from I_r and recovers the overflow values in e_r using O_2. Next, he decrypts E_d using Eqn. (11) to obtain the decrypted result e_d, recovers the overflow values in e_d using O_1, inverse scrambles e_d and e_r using the encryption key to obtain the decrypted estimation error values e which embedded with secret data. Finally, he recovers the original image using Eqn. (13). Note that this step may result in overflow, because the estimation error values are modified. We record these positions in a location map O_3 and embed it into the white set pixels using the traditional RDH method. Thus, the marked decrypted image I_m is generated.

If the receiver has data hiding key, he can further extract the secret data from the marked decrypted image. He first extracts O_3 from the white set pixels of I_m, recovers the overflow pixels using O_3, then calculates the estimation error using Eqn. (2). According to the encryption and sharing keys, the receiver scrambles the estimation error and separates it into two parts: $\tilde{e}_d$ which contains secret data and the remainder part e_r. Then, he extracts the encrypted secret data from $\tilde{e}_d$ using Eqn. (7), recovers $\tilde{e}_d$ using Eqns. (8) and (12). Finally, he inverse scrambles the recovered error values and obtains the original image using Eqn. (13).

3. SIMULATION RESULTS AND COMPARISONS

We use the standard test image *Lena* selected from the Miscelaneous database[1] for simulation and the results are shown in Fig. 3. With the embedding rate = 0.02 bpp, the marked decrypted image has a high image quality of 59.916 dB compared with the original image. The recovered image in Fig. 3(d) is exactly the same with the original image due to the reversibility of the proposed method.

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

Fig. 3: (a) the original image; (b) the marked encrypted image with embedding rate = 0.02 bpp; (c) the marked decrypted image with PSNR=59.916 dB and (d) the recovered image.

We select two images which are commonly used by other RDHEI methods from the Miscelaneous database[1] to show the quality of the marked decrypted image generated by proposed method and other RDHEI methods. The results are plotted in Fig. 4. From the results we can obvious that our proposed method achieves significantly higher PSNRs than those RDHEI methods in [2–4, 7], which encrypt the original image without preprocessing. Compared with Zhang's method in [9], our proposed method significantly improves the embedding rate and slightly improves the PSNRs.

We randomly selected 500 images in BOWSBase[2] to show the average maximum embedding rate of proposed algorithm and the method in [9]. The results are listed in Table 1. From the results we can observe that our proposed method can reach significantly higher embedding rate than the method in [9].

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

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

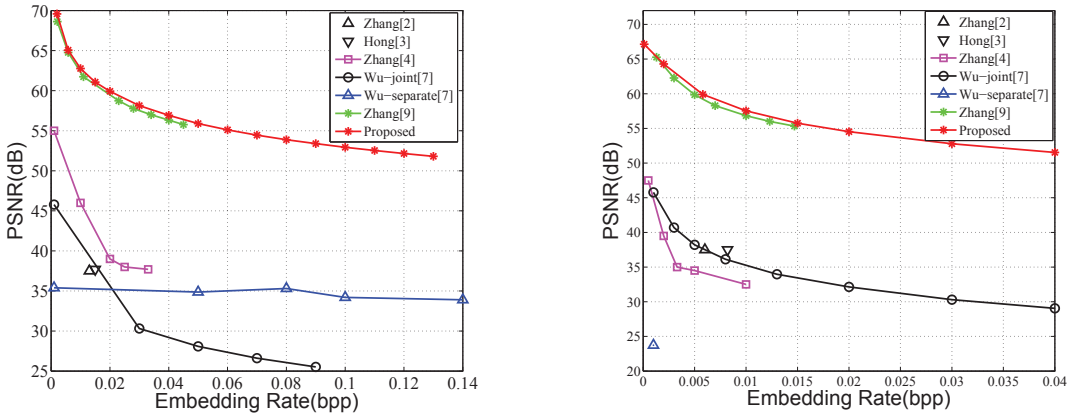

Fig. 4: PSNR comparisons of the marked decrypted images generated by proposed and other RDHEI methods. (a) *Lena*; (b) *Baboon*.

(.bpp)	**Zhang *et al*'s [9]**	**Proposed**
Average maximum embedding rate	0.0627	0.2386

Table 1: Average maximum embedding rate on 500 images of BOWSBase[2].

4. CONCLUSION

This paper has proposed an improved version of the RDHEI method in [9]. We estimate half of the pixels in the original image to obtain the estimation error values for embedding the secret data, so that the maximum embedding rate can be significantly improved. Experimental results has proved that our proposed method outperforms other relative methods both in PSNR and embedding rate.

5. REFERENCES

[1] W. Puech, M. Chaumont, and O. Strauss, "A reversible data hiding method for encrypted images," *Security, Forensics, Steganography, and Watermarking of Multimedia Contents X, Proceedings of SPIE 6819*, 2008.

[2] X. Zhang, "Reversible data hiding in encrypted image," *IEEE Signal Processing Letters*, vol. 18, no. 4, pp. 255–258, 2011.

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

[4] X. Zhang, "Separable reversible data hiding in encrypted image," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 2, pp. 826–832, 2012.

[5] X. Zhang, Z. Qian, G. Feng, and Y. Ren, "Efficient reversible data hiding in encrypted images," *Journal of Visual Communication and Image Representation*, vol. 25, no. 2, pp. 322–328, 2014.

[6] Z. Qian, X. Han, and X. Zhang, "Separable reversible data hiding in encrypted images by n-nary histogram modification," *The Third International Conference on Multimedia Technology*, p. 8, 2013.

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

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

[9] W. Zhang, K. Ma, and N. Yu, "Reversibility improved data hiding in encrypted images," *Signal Processing*, vol. 94, no. 0, pp. 118–127, 2014.

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