A Reserving Room Approach for Reversible Data Hiding Algorithm before Encryption on Encrypted Digital Images

Nimse Madhuri S  
E&TC Department, NMU

Prof.P.M.Mahajan  
E&TC Department, NMU

Prof.D.D.Ahire  
E&TC Department, Pune University

Abstract— In this paper a novel framework for data hiding based on reversible data hiding is presented for lossless data recovery approach. Reversible data hiding on encrypted images is most successful approach for its excellent property of lossless data recovery. Data hiding in digital images is a challenging task from last few decades since maintaining the image contents confidentiality and security to hidden data is an area of concern. The conventional algorithms in the literature which are proposed are subjected to errors at data extraction or data restoration process because these algorithms embed data by reversibly vacating room from the encrypted images. The proposed algorithm is more efficient than the conventional algorithm to propose a novel method by reserving room before encryption with a traditional RDH algorithm, and thus it is easy for the data hider to reversibly embed data in the encrypted image. The proposed method can achieve real reversibility, that is, data extraction and image recovery are free of any error. Finally the Experiments show that this novel method can embed more than 10 times as large payloads for the same image quality as the previous methods.

Keywords— reversible data hiding, histogram shift, image encryption privacy protection, data hiding

I. INTRODUCTION

Reversible data hiding (RDH) approach in image processing is an innovative technique, where the information related to original cover recovered lossless algorithm, this lossless data extraction is done once the extraction of embedded message is successfully completed. The applications related to reversible data hiding are medical imagery, military imagery and law forensics, where no distortion of the original contents is allowed. In literature, Reversible data hiding is always a interesting area of research for many researchers because of its excellent of recovering the information without any loss. A researcher named Kaifker et al [1] firstly introduced rate distortion model approach for reversible data hiding algorithm which attracts many researchers, since this rate distortion approach proposes a recursive code construction model which yields the data in lossless manner. After some years a researcher named Zhang et al [18] emptied out space for additional data by finding syndromes of a check matrix, and the side information used at the decoder side is also the spatial correlation of decrypted images. All the respective encrypted image is divided into three blocks. The half portion of each block in flipped by three LSB’s, room can be vacated for the embedded bit. The data extraction and image recovery proceed by finding which part has been flipped in one block. This process can be realized with the help of spatial correlation in decrypted image. Hong et al.[17] amend Zhang’s method at the decoder side by further exploiting the spatial correlation using a different estimation equation and side match technique to achieve much lower error rate. These two methods mentioned above rely on spatial correlation of original image to extract data. That is, the encrypted image should be decrypted first before data extraction. To separate the data extraction from image decryption, Zhang [18] emptied out space for data embedding following the idea of compressing encrypted images [14], [15]. Compression of encrypted data can be formulated as source coding with side information at the decoder [14], in which the typical method is to generate the compressed data in lossless manner by exploiting the syndromes of parity-check matrix of channel codes.

The method in [18] compressed the encrypted LSBs to vacate room for additional data by finding syndromes of a parity-check matrix, and the side information used at the receiver side is also the spatial correlation of decrypted images. All the three methods try to vacate room from the encrypted images directly. However, since the entropy of encrypted images has been maximized, these techniques can only achieve small payloads [16], [17] or generate marked image with poor quality for large payload [18] and all of them are subject to some error rates on data extraction and/or image restoration. In the present paper, we propose a novel method for RDH in encrypted images, for which we do not “vacate room after encryption” as done in [16]–[18], but “reserve room before encryption”. Not only does the proposed method separate data extraction from image decryption but also achieves excellent performance in two different prospects:
• Real reversibility is realized, that is, data extraction and image recovery are free of any error.
• For given embedding rates, the PSNRs of decrypted image containing the embedded data are significantly improved; and for the acceptable PSNR, the range of embedding rates is greatly enlarged.

III. PROPOSED WORK

Though losslessly vacating room from the encrypted images is bit difficult and sometimes inefficient, why are we still so possessed to find novel RDH techniques working for encrypted images? If we reverse the order of encryption and vacating room, i.e., reserving room prior to image encryption at content owner side, the RDH tasks in encrypted images would be more natural and much easier which leads us to the novel framework, “reserving room before encryption (RRBE)”. As shown in Fig.1 (b), the content owner first reserves enough space on original image and then convert the image into its encrypted version with the encryption key. Now, the data embedding process in encrypted images is inherently reversible for the data hider only needs to accommodate data into the spare space previous emptied out. The data extraction and image recovery are identical to that of Framework VRAE. Obviously, standard RDH algorithms are the ideal operator for reserving room before encryption and can be easily applied to Framework RRBE to achieve better performance compared with techniques from Framework VRAE. This is because in this new framework, we follow the customary idea that first losslessly compresses the redundant image content (e.g., using excellent RDH techniques) and then encrypts it with respect to protecting privacy.

![Diagram](image-url)

Fig. 1. (a) Framework VRAE. (b) Framework RRBE.

Next, we elaborate a practical method based on the Framework “RRBE”, which primarily consists of four stages: generation of encrypted image, data hiding in encrypted image, data extraction and image recovery. Note that the reserving operation we adopt in the proposed method is a traditional RDH approach.

(1) Generation of Encrypted Digital Images

To construct the encrypted image, the first stage can be divided into three main steps: image partition, self-reversible embedding followed by image encryption. At the beginning, image partition step divides original image into two parts A and B; then, the LSBs of A are reversibly embedded into B with a standard RDH algorithm so that LSBs of A can be used for accommodating messages; at last, encrypt the rearranged image to generate its encrypted version.
(a) Image Partition
The operator here for reserving room before encryption is a standard RDH technique, so the goal of image partition is to construct a smoother area B, on which standard RDH algorithms such as [10], [11] can achieve better performance. To do that, without loss of generality, assume the original image C is an 8 bits gray-scale image with its size M*N and pixels $C_{i,j}$, $0 \leq i \leq M, 1 \leq j \leq N$. First, the content owner extracts from the original image C, along the rows, several overlapping blocks whose number is determined by the size of to-be-embedded messages, denoted by $m$. In detail, every block consists of $m$ rows, where $m= \lceil 1/n \rceil$, and the number of blocks can be computed through $n=M-m+1$. An important point here is that each block is overlapped by $n/2$ rows and/or sub sequential blocks along the rows. For each block, define a function to measure its first-order smoothness

$$f = \sum_{i=2}^{m} \sum_{j=2}^{N-1} \left| C_{i,j} - C_{i,j-1} + C_{i+1,j} - C_{i,j+1} \right| \frac{1}{4} \tag{1}$$

Higher $f$ relates to blocks which contain relatively more complex textures.

(b) Self Reversible Embedding
The goal of self-reversible embedding is to embed the LSB-planes of A into B by employing traditional RDH algorithms. For illustration, we simplify the method in [10] to demonstrate the process of self-embedding. Note that this step does not rely on any specific RDH algorithm. Pixels in the rest of image B are first categorized into two sets: white pixels with its indices $i$ and $j$ satisfying $(i+j) \mod 2=0$ and black pixels whose indices meet $(i+j) \mod 2=1$, as shown in Fig. 2. Then, each white pixel $B_{i,j}$ is particular block with the highest $f$ to be A, and puts it to the front of the image concatenated by the rest part with fewer textured areas, as shown in Fig. 2.

![Image Partition and Embedding Process](image)

Estimated by the interpolation value obtained with the four black pixels surrounding it as follows

$$B_{i,j}' = w_0 B_{i-1,j} + w_1 B_{i+1,j} + w_2 B_{i,j-1} + w_3 B_{i,j+1} \tag{2}$$

Where the weight $w_i, 1 \leq i \leq 4$, is determined by the same method as proposed in [10]. The estimating error is calculated via $e_{i,j}=B_{i,j}'-B_{i,j}$ and then some data can be embedded into the estimating error sequence with histogram shift, which will be described later.

(c) Image Encryption
After rearranged self-embedded image, denoted by X, is generated, we can encrypt X to construct the encrypted image, denoted by E. With a stream cipher, the encryption version of X is easily obtained. For example, a gray value $X_{i,j}$ ranging from 0 to 255 can be represented by 8bits, such that $X_{i,j}(0), X_{i,j}(1), X_{i,j}(2)\ldots.. X_{i,j}(7)$

$$X_{i,j}(k) = \left[ \frac{X_{i,j}}{2^k} \right] \mod 2, \quad k = 0, 1, \ldots, 7 \tag{3}$$

The encrypted bits $E_{i,j}(k)$ can be calculated through exclusive-or operation

$$E_{i,j}(k) = X_{i,j}(k) \oplus \tau_{i,j}(k) \tag{4}$$

Where $\tau_{i,j}(k)$ is generated via a standard stream cipher determined by the encryption key. Finally, we embed 10 bits information into LSBs of first 10 pixels in encrypted version of A to tell data hider the number of rows and the number of bit-planes he can
embed information into. Note that after image encryption, the data hider or a third party cannot access the content of original image without the encryption key, thus privacy of the content owner being protected.

(2) Data Hiding in Encrypted Image

Once the data hider acquires the encrypted image E, he can embed some data into it, although he does not get access to the original image. The embedding process starts with locating the encrypted version of A, denoted by A_E. Since A_E has been rearranged to the top of E, it is effortless for the data hider to read 10 bits information in LSBs of first 10 encrypted pixels. After knowing how many bit-planes and rows of pixels he can modify, the data hider simply adopts LSB replacement to substitute the available bit-planes with additional data m. Finally, the data hider sets a label following m to point out the end position of embedding process and further encrypts m according to the data hiding key to formulate marked encrypted image denoted by E'. Anyone who does not possess the data hiding key could not extract the additional data.

(3) Data encryption and image recovery

Since data extraction is completely independent from image decryption, the order of them implies two different practical applications. 1) Case 1: Extracting Data from Encrypted Images: To manage and update personal information of images which are encrypted for protecting clients’ privacy, an inferior database manager may only get access to the data hiding key and have to manipulate data in encrypted domain. The order of data extraction before image decryption guarantees the feasibility of our work in this case. When the database manager gets the data hiding key, he can decrypt the LSB-planes of and extract the additional data by directly reading the decrypted version. When requesting for updating information of encrypted images, the database manager, then, updates information through LSB replacement and encrypts updated information according to the data hiding key all over again. As the whole process is entirely operated on encrypted domain, it avoids the leakage of original content.

2) Case 2: Extracting Data from Decrypted Images: In Case 1, both embedding and extraction of the data are manipulated in encrypted domain. On the other hand, there is a different situation that the user wants to decrypt the image first and extracts the data from the decrypted image when it is needed. The following example is an application for such scenario. Assume Alice out sourced her images to a cloud server, and the images are encrypted to protect their contents. Into the encrypted images, the cloud server marks the images by embedding some notation, including the identity of the images’ owner, the identity of the cloud server and time stamps, to manage the encrypted images. Note that the cloud server has no right to do any permanent damage to the images. Now an authorized user, Bob who has been shared the encryption key and the data hiding key, downloaded and decrypted the images. Bob hoped to get marked decrypted images, i.e., decrypted images still including the notation, which can be used to trace the source and history of the data. The order of image decryption before/without data extraction is perfectly suitable for this case. Next, we describe how to generate a marked decrypted image.

(a) Generating the marked Decrypted image

Step 1. With the encryption key, the content owner decrypts the image except the LSB-planes of A_E. The Decrypted version of E containing the embedded data can be calculated by

\[ X_l''(k) = E_l'(k) \oplus n_l(k) \quad (5) \]

\[ X_l'' = \sum_{l} X_l''(k) \times 2^k \quad (6) \]

Step 2. Extract the SR and ER in marginal area of B’. The plain image containing embedded data is obtained.

Fig 3. original image
The PSNR of w1 is: 29.89 dB

The actual embedding rate is 0.50 bpp

---

**TABLE I**

DIFFERENT PARAMETERS OF IMAGES

<table>
<thead>
<tr>
<th>SR.NO</th>
<th>IMAGE</th>
<th>MEAN</th>
<th>STANDARD DEVIATION</th>
<th>VARIANCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BOAT</td>
<td>129.706</td>
<td>46.67</td>
<td>137,439</td>
</tr>
<tr>
<td>2</td>
<td>AIRPLANE</td>
<td>181.909</td>
<td>44.09</td>
<td>355,376</td>
</tr>
<tr>
<td>3</td>
<td>LENA</td>
<td>123.545</td>
<td>47.855</td>
<td>1548.98</td>
</tr>
<tr>
<td>4</td>
<td>PEPPERS</td>
<td>119.70</td>
<td>53.87</td>
<td>133.15</td>
</tr>
</tbody>
</table>

---

**IV. CONCLUSIONS**

Reversible data hiding in encrypted images is a new topic drawing attention because of the privacy-protection requirements from large data management. Previously implementation of RDH in encrypted images by vacating room after encryption was as we did by reserving room before encryption. Here the data hider is benefited from the extra space emptied out in previous stage to make data hiding process easy. This proposed method gets advantage of all traditional RDH techniques for grayscale images and achieve excellent performance without loss of perfect data. Furthermore, this novel method achieves real reversibility, separate data extraction and improvement on the quality of marked decrypted images.
REFERENCES


