International Journal of Engineering Research
Volume No.3, Issue No.7, pp : 435-437

ISSN:2319-6890)(online),2347-5013(print)
01 July 2014

Reversible Data Hiding In Encrypted Images by Reserving Room before Encryption

Harish G, Smitha Shekar B, Prajwal R, Sunil S Shetty
Department of Computer Science & Engineering, Dr. Ambedkar Institute of Technology, India.
Email: c_harishg@yahoo.com, smithashekar_b@yahoo.co.in, prajwalr07@gmail.com and shettysunil1992@gmail.com .

Abstract

Recently, more and more attention is paid to reversible data hiding (RDH) in encrypted images, since it maintains the excellent property that the original cover can be losslessly recovered after embedded data is extracted while protecting the image content's confidentiality. All previous methods embed data by reversibly vacating room from the encrypted images, which may subject to some errors on data extraction and/or image restoration. Here, a novel method is proposed so as to reserve 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, i.e., data extraction and image recovery are free of any error.

***Keywords*: Reversible Data Hiding, image encryption, PSNR.**

I. Introduction

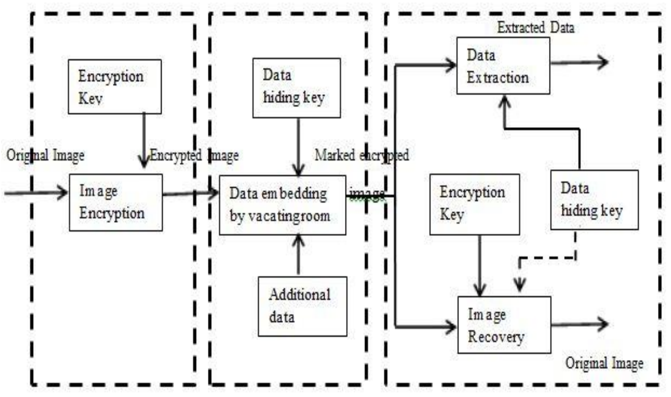
Reversible Data Hiding in images is a technique, by which the original cover can be losslessly recovered after the embedded message is extracted. This important technique is widely used in medical imagery, military imagery and law forensics, civil constructions where no distortion of the original cover is allowed.

For hiding data in an image: The method in [1] segments the encrypted image into a number of non-overlapping blocks; each block is used to carry one additional bit. The method [2] reduced the error rate of the method [1] by fully exploiting the pixels in calculating the smoothness of each block and using side match. The method in [3] compressed the encrypted LSBs to vacate room for additional data by finding syndromes of a parity-check matrix and to separate the data extraction from image decryption, emptied out space for data embedding following the idea of compressing encrypted images.

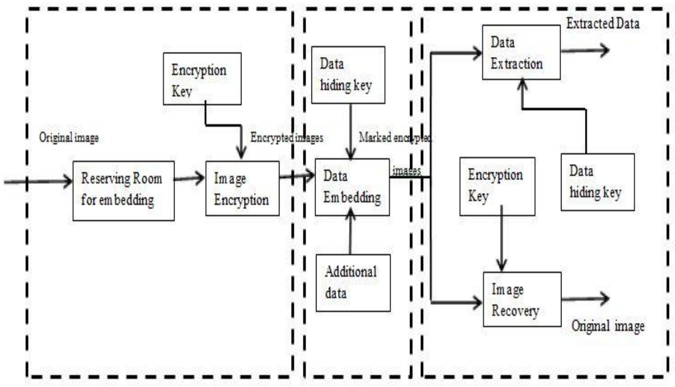
Here, a novel method is proposed so as to encrypt images using RDH , for which "vacate room after encryption" is not done in [1]–[3], but "reserve room before encryption"where, first empty out room by embedding LSBs of some pixels into other pixels with a traditional RDH method and then encrypt the image, so the positions of these LSBs in the encrypted image can be used to embed data which achieves excellent performance in two different prospects

• Real reversibility is realized, i.e., 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.



(a)



(b)

Figure 1 (a) Framework VRAE (b) Framework RRBE

II. Previous Methods

The methods proposed in [1]–[3] can be summarized asthe framework, "vacating room after encryption (VRAE)", as illustrated in Figure.1 (a). In this framework, a content owner encrypts the original image using a standard cipher with an encryption key. After producing the encrypted image, the content owner hands over it to a data hider (e.g., a database manager) and the data hider can embed some auxiliary data into the encrypted image by losslessly vacating some room according to a data hiding key. Then a receiver, may be the content owner himself or an authorized third party can extract the embedded data with the data hiding encrypted version according to the encryption key.

In all methods of [1]–[3], the encrypted 8-bit gray scale images are generated by encrypting every bit planes with a stream cipher. The method in [1] segments the encrypted image into a number of non-overlapping blocks sized by a×a, each block is used to carry one additional bit. To do this, pixels in each block are pseudo-randomly divided into two sets S1 and S2 according to a data hiding key. If the additional bit to be embedded is 0, flip the 3 LSBs of each encrypted pixel in S1, otherwise flip the 3 encrypted LSBs of pixels in S2. For data extraction and image recovery, the receiver flips all the three LSBs of pixels in S1 to form a new decrypted block, and flips all the three LSBs of pixels in S2 to form another new block; one of them will be decrypted to the original block. Due to spatial correlation in natural images, original block is presumed to be much smoother than interfered block and embedded bit can be extracted correspondingly. However, there is a risk of defeat of bit extraction and image recovery when divided block is relatively small or has much fine-detailed textures.

III. Proposed Method

Since losslessly vacating room from the encrypted images is relatively difficult and sometimes inefficient and reversing 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 to the novel framework, "Reserving Room Before Encryption (RRBE)". As shown in Figure. 1(b), the content owner first reserve enough space on original image and then converts 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 which 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, the customary idea is followed i.e., first losslessly compresses the redundant image content (e.g., using excellent RDH techniques) and then encrypts it with respect to protecting privacy. Next, 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, data extraction and image restoration.

A. Generation of Encrypted Image

Actually, to construct the encrypted image, the first stage can be divided into three 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 final version.

1) Image Partition: The reserving room before encryption is a standard RDH technique, so the goal of image partition is to construct a smoother area , on which standard RDH algorithms such as [4], [5] can achieve better performance. To do that, without loss of generality, assume the original image is an 8 bits gray-scale image with its size $M \times N$ and pixels $C_{i,j} \in [0,255], 1 \le i \le M \le j \le N$. First, the content owner extracts from the original image, along the rows, several overlapping blocks whose number is determined by the size of to be embedded messages, denoted by *l*. In detail, every block consists of *m* rows, where, *m=[l/N]*and the number of blocks can be computed through *n= M-m+1*. An important point here is that each block is overlapped by pervious and/or sub sequential blocks along the rows. For each block, define a function to measure its first-order smoothness

$$f = \sum_{u=2}^{m} \sum_{v=2}^{N-1} \left| \mathbf{C}_{u,v} - \frac{\mathbf{C}_{u-1,v} + \mathbf{C}_{u+1,v} + \mathbf{C}_{u,v-1} + \mathbf{C}_{u,v+1}}{4} \right| \quad (1)$$

Higher f relates to blocks which contain relatively more complex textures. The content owner, therefore, selects the 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 Figure. 2.

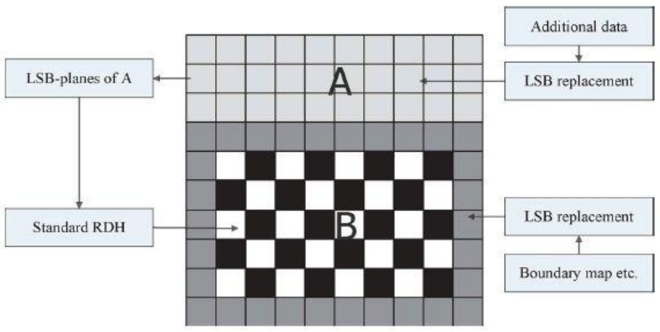


Figure.2 Illustration of image partition and embedding process.

2) 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, simplify the method in [4] 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*) *mod2= 0* and black pixels whose indices meet (*i + j*) *mod2 = 1*, as shown in Figure. 2. Then, each white pixel, $\mathbf{B}_{i,j}$, is estimated by the interpolation value obtained with the four black pixels surrounding it as follows

$$\mathbf{B}'_{i,j} = w_1\mathbf{B}_{i-1,j} + w_2\mathbf{B}_{i+1,j} + w_3\mathbf{B}_{i,j-1} + w_4\mathbf{B}_{i,j+1}, \quad (2)$$

where the weight $w_i, 1 \le i \le 4$, is determined by the same method as proposed in [4]. The estimating error is calculated via $e_{i,j} = \mathbf{B}_{i,j} - \mathbf{B}'_{i,j}$ and then some data can be embedded into the estimating error sequence with histogram shift, which will be described later. Further calculate the estimating errors of black pixels with the help of surrounding white pixels that may have been modified. Then another estimating error sequence is generated which can accommodate messages and can also implement multilayer embedding scheme by considering the modified **B** as "original" one when needed. In summary, to exploit all pixels of **B**, two estimating error sequences are constructed for embedding messages in every single-layer embedding process.

3) Image Encryption: After rearranged self-embedded image, denoted by **X** , is generated. Then 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 ranging from 0 to 255 can be represented by 8 bits, $\mathbf{X}_{i,j}(0), \mathbf{X}_{i,j}(1), \ldots, \mathbf{X}_{i,j}(7)$, such that

$$\mathbf{X}_{i,j}(k) = \left\lfloor \frac{\mathbf{X}_{i,j}}{2^k} \right\rfloor \bmod 2, \quad k = 0, 1, \ldots, 7. \quad (3)$$

The encrypted bits can be calculated through exclusive- or operation

$$\mathbf{E}_{i,j}(k) = \mathbf{X}_{i,j}(k) \oplus r_{i,j}(k), \quad (4)$$

Finally, embed 10 bits information into LSBs of first 10 pixels in encrypted version of to tell data hider the number of rows and the number of bit-planes that he can embed information into encrypted image. 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.

B. Data Hiding in Encrypted Image

Once the data hider acquires the encrypted image, 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 $\mathbf{A}_E$. Since $\mathbf{A}_E$ 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 $\mathbf{E}'$. Anyone who does not possess the data hiding key could not extract the additional data.

C. Data Extraction 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 $\mathbf{A}_E$ and extract the additional data **m** 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 outsourced her images to a cloud server, and the images are encrypted to protect their contents. In that encrypted images, the cloud server marks the images by embedding some notation, including the identity of the image 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.

IV. Conclusion

Reversible data hiding in encrypted images is a new topic drawing attention because of the privacy preserving requirements from cloud data management. Previous methods implement RDH in encrypted images by vacating room after encryption, as opposed to which reserving room before encryption is proposed. Thus the data hider can benefit from the extra space emptied out in previous stage to make data hiding process effortless. The proposed method can take advantage of all traditional RDH techniques for plain images and achieve excellent performance without loss of perfect secrecy. Furthermore, this novel method can achieve real reversibility, separate data extraction and greatly improvement on the quality of marked decrypted images.

REFERENCES

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

ii. *W. Hong, T. 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.*

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

iv. *L. Luo et al., "Reversible image watermarking using interpolation technique," IEEE Trans. Inf. Forensics Security, vol. 5, no. 1, pp. 187–193, Mar. 2010.*

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