Hide Inside-Separable Reversible Data Hiding in Encrypted Image

Lalit Dhande, Priya Khune, Vinod Deore, Dnyaneshwar Gawade

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 memory space from the encrypted images, which may be subject to some errors on data extraction and/or image restoration. In this paper, we propose a novel method by reserving memory space before encryption with a traditional RDH technique, and thus it is easy for the data hider to reversibly embed data in the image. The proposed method can achieve real reversibility, that is, data extraction and image recovery are free of any error.

Index Terms— ***Data Encryption, Reversible Data Hiding, Image Encryption, Privacy Protection, Data Extraction.***

I. INTRODUCTION

Reversible data hiding (RDH) 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, where no distortion of the original cover is allowed. Since first introduced, RDH [9] has attracted considerable research interest. With regard to providing confidentiality for images, encryption is an effective and popular means as it converts the original and meaningful content to incomprehensible one. Although few RDH techniques in encrypted images have been published yet, there are some promising applications if RDH [9] can be applied to encrypted images. To separate the data extraction from image decryption, the idea of compressing encrypted images and the space for data embedding; Compression of encrypted data [1] can be formulated as source coding with side information at the decoder, 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 compressed the encrypted LSBs to memory space for additional data by finding syndromes of a parity-check Matrix [7] and the side information used at the receiver side is also the spatial correlation of decrypted images.

Manuscript received February, 2014.

Mr. Lalit Dhande, Computer Department, Pune University/ Parvatibai Genba Moze College of Engineering, Pune, India.

Prof. Mrs. Priya Khune, Computer Department, Pune University/ Parvatibai Genba Moze College of Engineering, Pune, India.

Mr. Vinod Deore, Computer Department, Pune University/ Parvatibai Genba Moze College of Engineering, Pune, India.

Mr. Dnyaneshwar Gawade, Computer Department, Pune University/ Parvatibai Genba Moze College of Engineering, Pune, India.

In the present paper, we propose a novel method for RDH in encrypted images, for which we do not "vacate memory space after encryption", but "reserve memory space before encryption". In the proposed method, we first empty out memory space by embedding LSBs of some pixels into other pixels with a traditional RDH[9] method and then encrypt the image, so the positions of these LSBs in the encrypted image can be used to embed data. 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 [2], 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.

II. LITERATURE SURVEY

A. *Existing System Problem Statement:*

The reversible data hiding in encrypted image is investigated. Most of the work on reversible data hiding focuses on the data embedding/extracting on the plain spatial domain [8]. But, in some applications, an inferior assistant or a channel administrator hopes to append some additional message, such as the origin information, image notation or authentication data, within the encrypted image though he does not know the original image content.

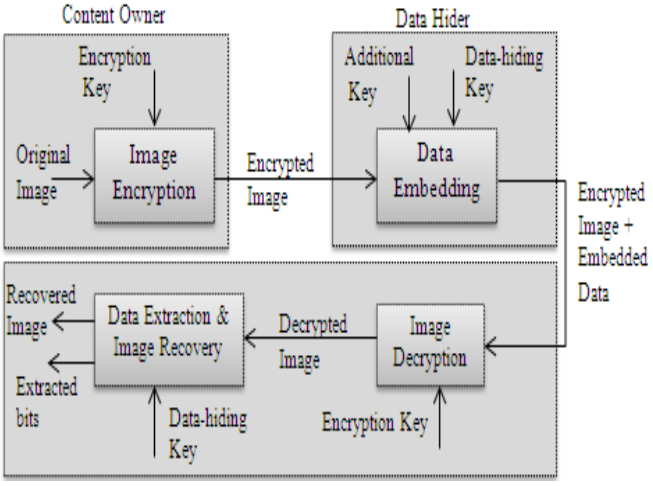

Fig.1 Non-Separable Reversible Data Hiding in Encrypted Image

And it is also hopeful that the original content should be recovered without any error after image decryption and message extraction at receiver side. Fig. 1 gives the sketch. A content owner encrypts the original image using an

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.

encryption key, and a data-hider can embed additional data into the encrypted image using a data-hiding key though he does not know the original content. With an encrypted image containing additional data, a receiver may first decrypt it according to the encryption key, and then extract the embedded data and recover the original image according to the data-hiding key. In the scheme, the data extraction is not separable[4] from the content decryption. In other words, the additional data must be extracted from the decrypted image, so that the principal content of original image is revealed before data extraction, and, if someone has the data-hiding key but not the encryption key, he cannot extract any information from the encrypted image containing additional data.

In the scheme, the original image is encrypted using an encryption key and the additional data are embedded into the encrypted image using a data-hiding key. With an encrypted image containing additional data, if the receiver has only the data-hiding key, he can extract the additional data though he does not know the image content. If he has only the encryption key, he can decrypt the received data to obtain an image similar to the original one, but cannot extract the embedded additional data. If the receiver has both the data-hiding key and the encryption key, he can extract the additional data and recover the original image without any error when the amount of additional data is not too large.

Since losslessly memory space from the encrypted images is relatively difficult and sometimes inefficient [1], why are we still so obsessed to find novel RDH techniques working directly for encrypted images? If we reverse the order of encryption and memory space , i.e., reserving memory space 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 memory space before encryption (RRBE)"[8].

This paper proposes a novel scheme for separable reversible data hiding in encrypted image. As shown in Fig. 2, 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 only needs to accommodate data into the spare space previous emptied out[8]. The data extraction and image recovery are identical to that of Framework VRAE. Obviously, standard RDH algorithms [9] are the ideal operator for reserving memory space 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. 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 [9].

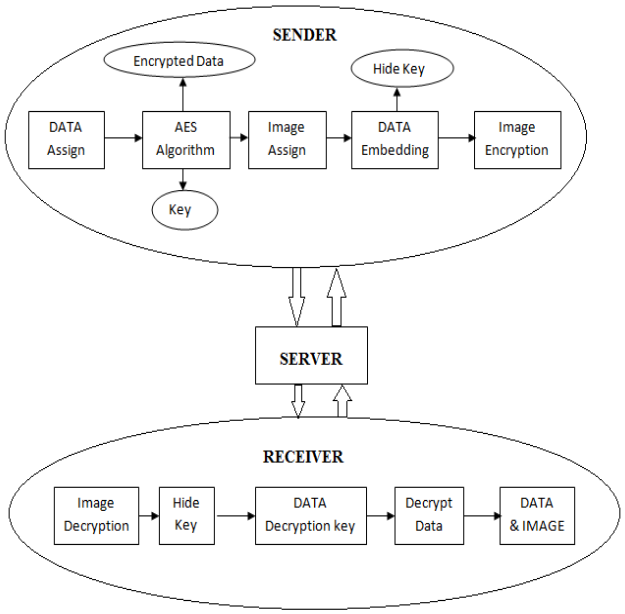

Fig.2 Separable Reversible Data Hiding in Encrypted Image

The proposed scheme is made up of image encryption, data embedding and data-extraction/ image-recovery phases. The content owner encrypts the original uncompressed image using an encryption key to produce an encrypted image. Then, the data-hider compresses the least significant bits (LSB) of the encrypted image using a data-hiding key to create a sparse space to accommodate the additional data [8]. At the receiver side, the data embedded in the created space can be easily retrieved from the encrypted image containing additional data according to the data-hiding key. Since the data embedding only affects the LSB, a decryption with the encryption key can result in an image similar to the original version. When using both of the encryption and data-hiding keys, the embedded additional data can be successfully extracted and the original image can be perfectly recovered by exploiting the spatial correlation in natural image.

B. Image Encryption

The user will browse the image from computer and encrypt the image and system will auto generate encryption key.

C. Data Encryption

The user will browse data that he want send and encrypt the original data and system will auto generate the data encryption key.

D. Data Embedding

User will hide the encrypted data in encrypted image and system will auto generate data hiding key and system will generate file extension as per user defined.

In the data embedding phase, some parameters are embedded into a small number of encrypted pixels, and the LSB of the other encrypted pixels are compressed to create a space for accommodating the additional data and the original data at the positions occupied by the parameters. The detailed procedure is as follows According to a data-hiding key, the data-hider randomly selects Np encrypted pixels that will be used to carry the parameters for data hiding [8]. Here, Np is a small positive integer, for example, Np=20. The other (N-Np) encrypted pixels are permuted and divided into a

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.

number of groups, each of which contains L pixels. The permutation way is also determined by the data-hiding key. For each pixel-group, collect the M least significant bits of the L pixels, and denote them as B (k,1) , B (k,2) …… B(k,M*L) where k is a group index within [1,(N-Np)/L] and M is a positive integer less than 5.The data-hider also generates a matrix G, which is composed of two parts. The left part is the identity matrix and the right part is pseudo-random binary matrix derived from the data-hiding key. For each group, which is product with the G matrix to form a matrix of size (M * L-S). Which has sparse bits of size S, in which the data is embedded and arrange the pixels into the original form and permutated to form a original image.

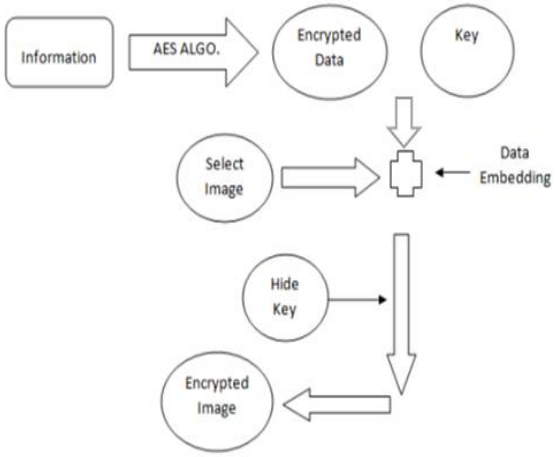

Fig. 3 Data Embedding Process

When having an encrypted image containing embedded data, a receiver firstly generates ri, j, k according to the encryption key, and calculates the exclusive-or of the received data and ri, j, k to decrypt the image. We denote the decrypted bits as bri, j and k. Clearly, the original five most significant bits (MSB) are retrieved correctly. For a certain pixel, if the embedded bit in the block including the pixel is zero and the pixel belongs to S1, or the embedded bit is 1 and the pixel belongs to S0, the data-hiding does not affect any encrypted bits of the pixel. So, the three decrypted LSB must be same as the original LSB, implying that the decrypted gray value of the pixel is correct [5]. On the other hand, if the embedded bit in the pixel's block is 0 and the pixel belong to S0, or the embedded bit is 1 and the pixel belongs to S1which is decrypted LSB[5]. This project proposes a novel scheme for separable reversible data hiding in encrypted image. In the proposed scheme, the original image is encrypted using an encryption key and the additional data are embedded into the encrypted image using a data-hiding key. With an encrypted image containing additional data, if the receiver has only the data-hiding key, he can extract the additional data though he does not know the image content. If he has only the encryption key, he can decrypt the received data to obtain an image similar to the original one, but cannot extract the embedded additional data. If the receiver has both the data-hiding key and the encryption key, he can extract the additional data and recover the original image without any error when the amount of additional data is not too large.

E. File Sending

The sender will send the file with any extension (user defined) using internet connection and all keys will send privately.

F. Image Decryption

After receiving file from user, if receiver have Data hiding and Image Encryption keys then he will only decrypt the image and he will not able to get original data.

G. Data Decryption

If receiver has data hiding and data encryption keys then he will able to decrypt data. After decryption of data he will get the original data.

H. Data Extraction & 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 encrypted version of A denoted by A_E, and extract the additional data m by directly reading the decrypted version [9]. 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 [9]. 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 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.

III. AES ALGORITHM

In our system we are using 128 bit key and in AES this is represented by Nb = 4, which reflects the number of 32-bit words (number of columns) in the State.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.

The length of the Cipher Key, K, is 128. The key length is represented by Nk = 4, 6, or 8, which reflects the number of 32-bit words (number of columns) in the Cipher Key [7].
The number of rounds to be performed during the execution of the algorithm is dependent on the key size. The number of rounds is represented by Nr, where Nr = 10 when Nk = 4, Nr = 12 when Nk = 6, and Nr = 14 when Nk = 8.
For both its Cipher and Inverse Cipher, the AES algorithm uses a round function that is composed of four different byte-oriented transformations:
1) Substitution using a substitution table (S-box).
2) Shifting rows of the State array by different offsets
3) Mixing the data within each column of the State array
4) Adding a Round Key to the State. [6].

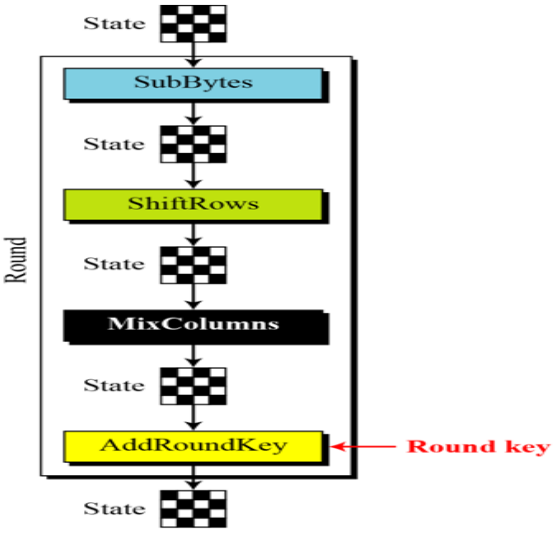

Fig.4 AES Round Function

IV. FUTURE SCOPE

The future implementation is to add support to hide all file formats. This allows for a much broader spectrum of usage: one would be able to encode .exe, .doc, .pdf, .mp3, etc. The system would be more versatile because often hiding text just isn't enough. We can also implement batch image processing and statistical analysis so that the system could run the code through a dataset of images and detect Steganography and perhaps crawl through Google Image Search to see how prevalent Steganography is.

A. Three Keys For More Data Security

Encrypted data is hidden in Encrypted Image with separate keys for Data Encryption, Data Hiding and Image Encryption. For decrypting of data receiver should have both Data Encryption and Data hiding key.

B. Protection For Auto Generated Keys

To perform any operation the user has only 3 attempts. If user is fail to perform any of operation means user enter wrong 3 times then the system is goes to not responding state and one mail with receiver computer IP address is send to the admin.

C. User Define Extension Prevent Hacker Attack

During data hiding process user has to give the extension like .xyz.

D. Admin as Main

Admin of system have all authority that is admin can block, unblock any user at any time if he feel something wrong and admin have all records from all user.

V. 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 memory space after encryption, as opposed to which we proposed by reserving memory space before encryption. Our study helps constructing secure transmission of secrete file preventing any third party access and security level of data is increased by encrypting data. We also provide protection for keys during decryption process if any hacker attacks on system. In future we can use audio, video in case of image as cover for hiding the data.

REFERENCES

[1] M. Johnson, P. Ishwar, V. M. Prabhakaran, D. Schonberg, and K. Ramchandran, "On compressing encrypted data," IEEE Trans. Signal Process., vol. 52, no.10, pp. 2992-3006, Oct. 2004.
[2] Z. Ni, Y.-Q. Shi, N. Ansari, and W. Su, "Reversible data hiding," IEEE Trans. Circuits Syst. Video Technol., vol. 16, no. 3, pp. 354-362, Mar.2006.
[3] C.-C. Chang, C.-C.Lin, and Y.-H. Chen, "Reversible data-embedding scheme using differences between original and predicted pixel values, "IET Inform. Security, vol. 2, no. 2, pp.35-46, 2008.
[4] T. Bianchi, A. Piva, and M. Barni, "On the implementation of the discrete Fourier transform in the encrypted domain," IEEE Trans. Inform. Forensics Security, vol. 4, no. 1, pp. 86-97, Feb. 2009.
[5] W. Liu, W. Zeng, L. Dong, and Q. Yao, "Efficient compression of encrypted gray scale images," IEEE Trans. Image Process., vol. 19, no. 4, pp. 1097-1102, Apr. 2010.
[6] T. Bianchi, A. Piva, and M. Barni, "Composite signal representation for fast and storage- efficient processing of encrypted signals," IEEE Trans. Inform. Forensics Security, vol. 5, no. 1, p. 180-187, Feb. 2010.
[7] X. Zhang, "Lossy compression and iterative reconstruction for encrypted image," IEEE Trans. Inform. Forensics Security, vol. 6, no.1, pp. 53-58, Feb. 2011.
[8] Xinpeng Zhang "Separable Reversible Data Hiding in Encrypted Image" IEEE Trans. VOL. 7, no. 2, Apr 2012.
[9] Kede Ma, Weiming Zhang, "Reversible Data Hiding in Encrypted Images by Reserving Room Before Encryption" IEEE Trans. VOL. 8, no. 3, Mar 2013.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.