IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 1, Ver. III (Jan. 2014), PP 08-12
www.iosrjournals.org

Ensuring Distributed Accountability for Data Sharing Using Reversible Data Hiding in Cloud Over-Lay Network

E. Kalaikavitha M.C.A.,(M.Phil.,)
Assistant Professor, Department of Information Technology, Rathinam College of Arts and Science , Eachanari, Pollachi Main Road, Coimbatore-641021, Tamil Nadu, India.

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 lossless 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 be subject to some errors on data extraction and/or image restoration. In this paper, we 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. A major feature of the centralized database services is that users' data are usually processed remotely in unknown machines that users do not own or operate. While enjoying the convenience brought by this new emerging technology, users' fears of losing control of their own data (particularly, financial and health data) can become a significant barrier to the wide adoption of centralized database services. To address this problem, in this paper, we propose a novel highly decentralized information accountability framework to keep track of the actual usage of the user's data in the cloud Over-lay Network. We leverage the LOG file create a dynamic and traveling object, and to ensure that any access to users' data will trigger authentication and automated logging local to the LOGs. To strengthen user's control, we also provide distributed auditing mechanisms. We provide extensive experimental studies that demonstrate the efficiency and effectiveness of the proposed approaches.*
Index Terms : *Reversible data hiding, image encryption, privacy protection, data sharing.*

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 has attracted considerable research interest. By first extracting compressible features of original cover and then compressing them losslessly, spare space can be saved for embedding auxiliary data. A more popular method is based on difference expansion (DE), in which the difference of each pixel group is expanded, e.g. multiplied by 2, and thus the least significant bits (LSBs) of the difference are all-zero and can be used for embedding messages. Another promising strategy for RDH is histogram shift (HS), in which space is saved for data embedding by shifting the bins of histogram of gray values. The state-of-art methods usually combined DE or HS to residuals of the image, e.g., the predicted errors, to achieve better performance.

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 can be applied to encrypted images. In Hwang *et al.* advocated a reputation-based trust-management scheme enhanced with data coloring (a way of embedding data into covers) and software watermarking, in which data encryption and coloring offer possibilities for upholding the content owner's privacy and data integrity. Obviously, the cloud service provider has no right to introduce permanent distortion during data coloring into encrypted data. Thus, a reversible data coloring technique based on encrypted data is preferred. Suppose a medical image database is stored in a data center, and a server in the data center can embed notations into an encrypted version of a medical image through a RDH technique. With the notations, the server can manage the image or verify its integrity without having the knowledge of the original content, and thus the patient's privacy is protected. On the other hand, a doctor, having the cryptographic key, can decrypt and restore the image in a reversible manner for the purpose of further diagnosing.

Some attempts on RDH in encrypted images have been made. In Zhang divided the encrypted image into several blocks. By flipping 3 LSBs of the half of pixels in each block, 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.* ameliorated 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 emptied out space for data embedding following the idea of compressing encrypted images. Compression of encrypted data 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 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.

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, but "reserve room before encryption". In the proposed method, we 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. 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.

II. Proposed Method

In this section, we present an overview of the centralized Information Accountability framework and discuss how the CIA framework meets the design requirements discussed in the previous section. The Centralized Information Accountability framework proposed in this work conducts automated logging and distributed auditing of relevant access performed by any entity, carried out at any point of time at any cloud service provider.

It has two major components: logger and log harmonizer.

2.1 Major Components

There are two major components of the CIA, the first being the logger, and the second being the log harmonizer. The logger is the component which is strongly coupled with the user's data, so that it is downloaded when the data are accessed, and is copied whenever the data are copied. It handles a particular instance or copy of the user's data and is responsible for logging access to that instance or copy. The log harmonizer forms the central component which allows the user access to the log files. The logger is strongly coupled with user's data (either single or multiple data items). Its main tasks include automatically logging access to data items that it contains, encrypting the log record using the public key of the content owner, and periodically sending them to the log harmonizer. It may also be configured to ensure that access and usage control policies associated with the data are honored.

For example, a data owner can specify that user X is only allowed to view but not to modify the data. The logger will control the data access even after it is downloaded by user X.

The logger requires only minimal support from the server (e.g., a valid Java virtual machine installed) in order to be deployed. The tight coupling between data and logger, results in a highly distributed logging system, therefore meeting our first design requirement.

Furthermore, since the logger does not need to be installed on any system or require any special support from the server, it is not very intrusive in its actions, thus satisfying our fifth requirement. Finally, the logger is also responsible for generating the error correction information for each log record and send the same to the log harmonizer. The error correction information combined with the encryption and authentication mechanism provides a robust and reliable recovery mechanism, therefore meeting the third requirement.

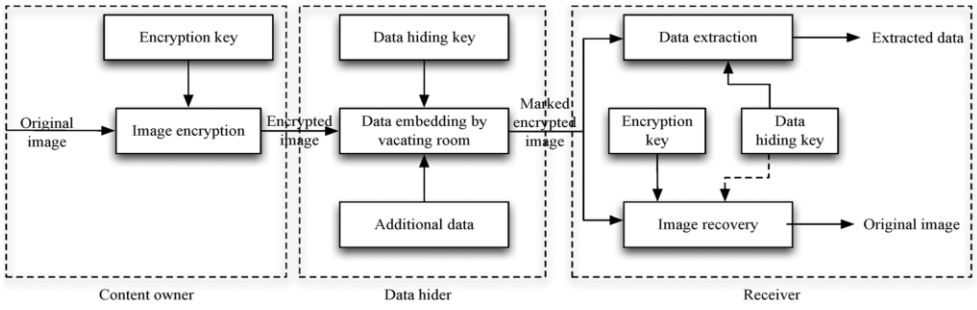
The log harmonizer is responsible for auditing. Being the trusted component, the log harmonizer generates the master key. It holds on to the decryption key for the IBE key pair, as it is responsible for decrypting the logs. Alternatively, the decryption can be carried out on the client end if the path between the log harmonizer and the client is not trusted. In this case, the harmonizer sends the key to the client in a secure key exchange.

It supports two auditing strategies: push and pull. Under the push strategy, the log file is pushed back to the data owner periodically in an automated fashion. The pull mode is an on-demand approach, whereby the log file is obtained by the data owner as often as requested. These two modes allow us to satisfy the aforementioned fourth design requirement. In case there exist multiple loggers for the same set of data items, the log harmonizer will merge log records from them before sending back to the data owner.

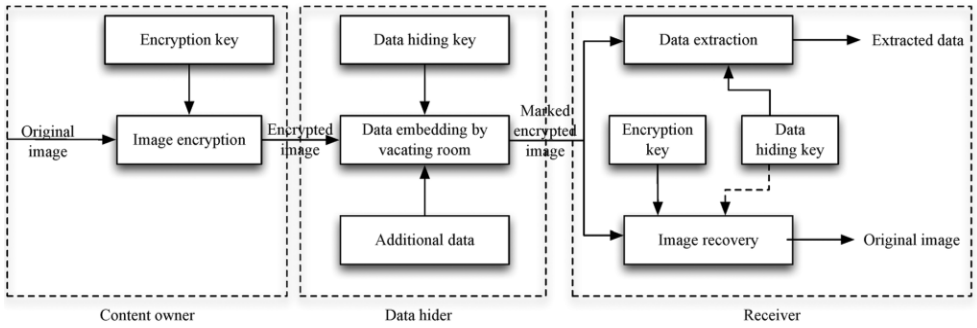
The log harmonizer is also responsible for handling log file corruption. In addition, the log harmonizer can itself carry out logging in addition to auditing. Separating the logging and auditing functions improves the performance. The logger and the log harmonizer are both implemented as lightweight and portable LOG files.

The LOG file implementation provides automatic logging functions, which meets the second design requirement. Since losslessly vacating room from the encrypted images is relatively difficult and sometimes inefficient, why are we still so obsessed to find novel RDH techniques working directly 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 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. 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



(a)

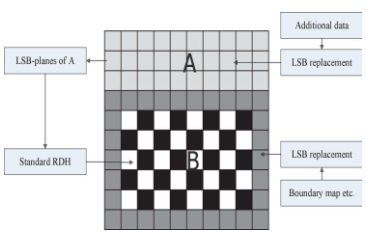


(a)

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.

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 Encryption:

After rearranged self-embedded image, denoted by , is generated, we can encrypts X to construct the encrypted image, denoted by .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 8 bits, $X_{i,i}(0)$, $X_{i,j}(1)$…. $X_{i,j}(7)$, such that

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

The encrypted bits $E_{i,j}(K)$…. can be calculated through exclusive or operation

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

Where $r_{i,j}(K)$ 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 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.

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.

2.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 A_E. Since A_E has been rearranged to the top of , 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 . 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^1 . Anyone who does not possess the data hiding key could not extract the additional data.

3.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 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. 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,

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

(a) Original image, (b) encrypted image, (c) decrypted image containing messages (embedding rate 0.1 bpp), (d) recovery version.

III. 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 we proposed by reserving room before encryption. 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.

We proposed innovative approaches for automatically logging any access to the data in the cloud over-lay network together with an auditing mechanism. Our approach allows the data owner to not only audit his content but also enforce strong back-end protection if needed. Moreover, one of the main features of our work is that it enables the data owner to audit even those copies of its data that were made without his knowledge.

References

[1] T. Kalker and F.M.Willems, "Capacity bounds and code constructionsfor reversible data-hiding," in *Proc. 14th Int. Conf. Digital Signal Processing (DSP2002)*, 2002, pp. 71–76.

[2] W. Zhang, B. Chen, and N. Yu, "Capacity-approaching codes for reversible data hiding," in *Proc 13th Information Hiding (IH'2011), LNCS 6958*, 2011, *West, Electronic Imaging, Security and Watermarking of Multimedia Contents*, San Jose, CA, USA, Jan. 2002, vol. 4675, pp. 572–583. pp. 255–269, Springer-Verlag.

[3] W. Zhang, B. Chen, and N. Yu, "Improving various reversible data hiding schemes via optimal codes for binary covers," *IEEE Trans.Image Process.*, vol. 21, no. 6, pp. 2991–3003, Jun. 2012.

[4] J. Fridrich and M. Goljan, "Lossless data embedding for all image formats," in *Proc. SPIE Proc. Photonics*

[5] J. Tian, "Reversible data embedding using a difference expansion," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 8, pp. 890–896, Aug. 2003.

[6] Z. Ni, Y. Shi, N. Ansari, and S. Wei, "Reversible data hiding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 3, pp. 354–362, Mar. 2006.

[7] D.M. Thodi and J. J. Rodriguez, "Expansion embedding techniques for reversible watermarking," *IEEE Trans. Image Process.*, vol. 16, no. 3, pp. 721–730, Mar. 2007.

[8] X. L. Li, B. Yang, and T. Y. Zeng, "Efficient reversible watermarking based on adaptive prediction-error expansion and pixel selection," *IEEE Trans. Image Process.*, vol. 20, no. 12, pp. 3524–3533, Dec. 2011.