Separable Reversible Data Hiding in Encrypted Video File

Gayatri Pisal[1] | Aparna More[2] | Sonali Pawar[3] | Ashwini Vishwakarma[4]
[1,2,3,4](Department of Computer Engineering, Savitribai Phule Pune University, Nashik, India)
(gayatripisal93@gmail.com[1], aparnamore39@gmail.com[2], sony223pawar4@gmail.com[3], ashwinivishwakarma16@gmail.com[4])

Abstract— *As now-a-days security is the major issue to be considered. The use of computer networks for data transmission has created the need for security. In the previous system there were drawbacks i.e. the cover file was decrypted with distortions in it. To overcome the limitation of previous work we proposed hiding cipher textual data in encrypted video and compression as solution. Here we are using technique of Reserving Room Before Encryption (RRBE) with which we can overcome the drawbacks of existing system and decrypt the cover without any distortion. In this the sender encrypts the video and data separately, hides the data in encrypted video using LSB technique, while system auto generates the two respective keys. Receiver will need both the keys to extract the data. First, the encryption key is used for decrypting the video then using data hiding key the original data can be extracted.*

Keywords— Reversible data hiding, video encryption, privacy protection, RSA algorithm, LSB algorithm

I. INTRODUCTION

Reversible data hiding (RDH) in video 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.

In practical aspect, many RDH techniques have emerged in recent years. Fridrich *et al.* [4] constructed a general framework for RDH. 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) [5], 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.

The purpose of steganography is to hide the very presence of communication by embedding message into innocuous-looking cover objects. Also as an elective and popular means for privacy protection, encryption converts the ordinary signal into unintelligible data, so that the traditional signal processing usually takes place before encryption and after decryption.

The proposed system suggests that sender encrypts data and the original uncompressed video using the RSA algorithm and then after encryption, system will auto generate the data hiding and video encryption keys. Then user compresses the LSBs of the encrypted video technique and then user hides the data into encrypted video.

This paper is organized in the following manner. Section I is an Introduction, section II covers the description and drawbacks of proposed system, section III is the detailed description of proposed system with algorithms used, section IV is conclusion and section V contains references.

2.EXISTING SYSTEM

The methods proposed previously can be summarized as the framework, “vacating room after encryption (VRAE)”.

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 key and further recover the original image from the encrypted version according to the encryption key.

In all previous methods, the encrypted 8-bit gray-scale images are generated by encrypting every bit-plane with a stream cipher. The previous method segments the encrypted image into a number of non overlapping blocks sized by $\alpha \times \alpha$; 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 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 into form another new block; one of them will be decrypted to the original block. Due to spatial correlation in S2 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 (e.g. $\alpha=8$) or has much fine detailed textures.

3.PROPOSED SYSTEM

In Existing system losslessly vacating room from the encrypted videos is relatively difficult and sometimes inefficient, why are we still so obsessed to find novel RDH techniques working directly for encrypted videos? If we reverse the order of encryption and vacating room, i.e., reserving room prior to encryption at content owner side, the RDH tasks in encrypted video would be more natural and much easier which leads us to the novel framework, "reserving room before encryption(RRBE)", as illustrated in Fig. 1.

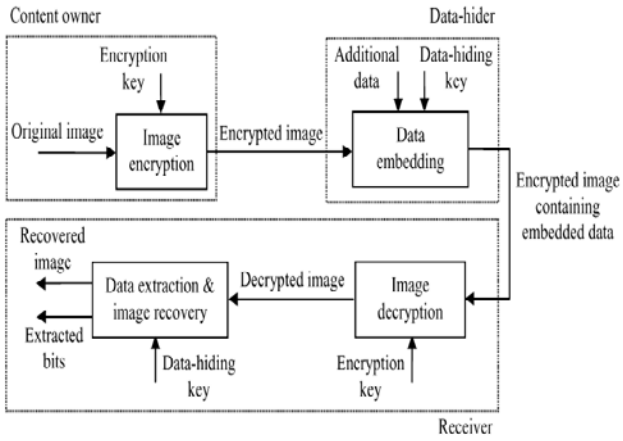

Fig.1 Block diagram of existing system

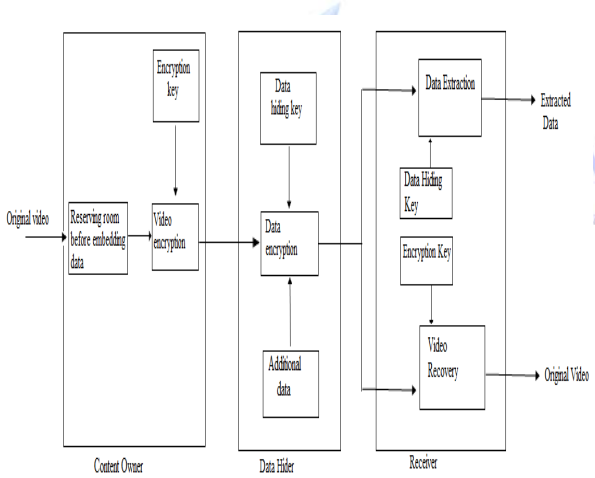

Fig.2 Block diagram of proposed system

The content owner first reserve enough space on original video and then converts the video into its encrypted version with the encryption key. Now, the data embedding process in encrypted video is inherently reversible for the data hider only needs to accommodate data into the spare space previous emptied out. The data extraction and video 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 video content (e.g., using excellent RDH techniques) and then encrypts it with respect to protecting privacy.

A. 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 video 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. Then, each white pixel, Bi,j, is estimated by the interpolation value obtained with the four black pixels surrounding it as follows:

$$\dot{B}i,j = w1\ B\ i-1,j + w2\ B\ i+1,j + w3\ B\ i,j-1 + w4\ B\ i,j+1 \quad (2)$$

where the weight wi, $1 \le i \le 4$, is determined by the same method as proposed in [10]. The estimating error is calculated via E i,j = Bi,j - $\dot{B}$i,j and then some data can be embedded into the estimating error sequence with histogram shift, which will be described later. After that, we 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 as well. Furthermore, we 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.

B. Video Encryption: After rearranged self-embedded video, denoted by X , is generated, we can encrypts X to construct the encrypted video, denoted by E. With a stream cipher, the encryption version of X is easily obtained. For example, a gray value Xi,j ranging from 0 to 255 can be represented by 8 bits, Xi,j(0), Xi,j(1),…,Xi,j(7), such that

$$Xi,j(k) = [Xi,j/\ 2\hat{}k] \bmod 2$$
Where k=0,1,…,7.

The encrypted bits Ei,j(k) can be calculated through exclusive-or operation

$$Ei,j(k) = Xi,j^{l}(k) \ominus Ri,j(k)$$

where Ri,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 video encryption, the data hider or a third party cannot access the content of original video without the encryption key, thus privacy of the content owner being protected.

C. Data Extraction and Video Recovery:

Since data extraction is completely independent from video decryption, the order of them implies two different practical applications.

Case 1: Extracting Data from Encrypted Video:

To manage and update personal information of video 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 video 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 videos, 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.

Case 2: Extracting Data From Decrypted Video:

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 video first and extracts the data from the decrypted video when it is needed. The following example is an application for such scenario. Assume Alice outsourced her videos to a cloud server, and the videos are encrypted to protect their contents. Into the encrypted videos, the cloud server marks the videos by embedding some notation,
including the identity of the videos owner, the identity of the cloud server and time stamps, to manage the encrypted videos. Note that the cloud server has no right to do any permanent damage to the videos. Now an authorized user, Bob who has been shared the encryption key and the data hiding key, downloaded and decrypted the videos. Bob hoped to get marked decrypted
videos, i.e., decrypted videos still including the notation, which can be used to trace the source and history of the data. The order of video decryption before/without data extraction is perfectly suitable for this case.

1)RSA:-

RSA stands for Ron Rivest, Adi Shamir and Leonard Adleman. RSA is an algorithm used by modern computers to encrypt and decrypt messages.

RSA involves a public key and private key. The public key can be known to everyone, it is used to encrypt messages. Messages encrypted using the public key can only be decrypted with the private key. The keys for the RSA algorithm are generated the following way:

1. Choose two different large random prime numbers ***p*** and ***q***.
2. Calculate ***n=pq***
   - ***n*** is the modulus for the public key and the private keys
3. Calculate the quotient: ***Ø(n)=(p-1)(q-1).***
4. Choose an integer ***e*** such that 1< ***e*** < ***Ø(n)***and ***e*** is coprime to ***Ø(n)*****i.e:** ***e*** and ***Ø(n)***share no factors other than 1; gcd(***e***,***Ø(n)***) = 1.
   - ***e*** is released s the public key exponent
5. Compute ***d*** to satisfy the congruence relation ***de≡1(modØ(n))*** **i.e.:** ***de=1+kØ(n)***
   for some integer ***k***.
   - ***d*** is kept as the private key exponent.

The public key is made of the modulus ***n*** and the public (or encryption) exponent ***e.*** The private key is made of the modulus ***n*** and the private (or decryption) exponent ***d*** which must be kept secret.

- For efficiency a different form of the private key can be stored:
  - ***p*** and ***q***: the primes from the key generation,
  - ***d mod (p-1)*** and ***d mod (q-1)***: often called *dmp1* and *dmq1*.
  - $q^{-1} mod\ (p)$: often called *iqmp*
- All parts of the private key must be kept secret in this form. ***p*** and ***q*** are sensitive since they are the factors of ***n***, and allow computation of ***d*** given ***e***. If ***p*** and ***q*** are not stored in this form of the private key then they are securely deleted along with other intermediate values from key generation.

2)LSB:-

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 , on which standard RDH algorithms such as [10], [11] 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*N and pixels
Ci,j ϵ [0,255] , 1≤ i ≤ M, 1 ≤ j ≤ 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 . In detail, every block consists of 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

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

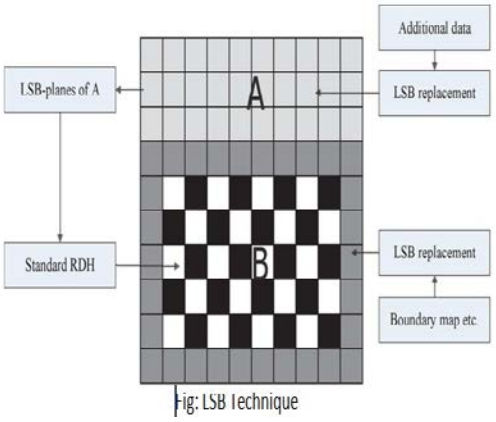

Fig: LSB Technique

Higher f relates to blocks which contain relatively more complex textures. The content owner, therefore, selects the particular block with the highest to be , and puts it to the front of the image concatenated by the rest B part with fewer textured areas, as shown in Fig. The above discussion implicitly relies on the fact that only single LSB-plane of A is recorded. It is straightforward that the content owner can also embed two or more LSB-planes of A into , which leads to half, or more than half, reduction in size of . However, the performance of , in terms of PSNR, after data embedding in the second stage decreases significantly with growing bit-planes exploited. Therefore, in this paper, we investigate situations that at most three LSB-planes of A are employed and determine the number of bit-plane with regard to different payloads experimentally in the next section.

4. Conclusion

Reversible data hiding in encrypted videos is a new topic drawing attention because of the privacy-preserving requirements from cloud data management. Previous methods implement RDH in encrypted videos 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 videos 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 videos.

References

[1] Vikash Ramesh, Kaushik Narayanan, Premlatha Pandian "Steganography in Audio Signals using Variable Bit Replacement Method in DCT domain" International Journal of Engineering Research and Technology(IJERT) Vol no.:3, Issue: 4, April 2014.

[2] Kede Ma, Weiming Zhang, Xianfeng Zhao, Member, IEEE, Nenghai Yu, and Fenghua Li, TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, VOL. 8, NO. 3, MARCH 2013.
"Reversible Data Hiding in Encrypted Images by Reserving Room Before Encryption ", IEEE

[3] Xinpeng Zhang, "Separable Reversible Data Hiding in Encrypted Image", IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, VOL. 7, NO. 2, APRIL 2012

[4] X. Zhang, "Lossy compression and iterative reconstruction for encrypted image," IEEE Trans. Inform. Forensics Security, vol. 6, no. 1, pp. 53–58, Feb. 2011.

[5] W. Liu, W. Zeng, L. Dong, and Q. Yao, "Efficient compression of encrypted
grayscale images," IEEE Trans. Image Process., vol. 19, no. 4, pp. 1097–1102, Apr. 2010.

[6] 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.

[7] Marghny Mohamed, Fadwa Al-Afari, Mohamed Bamatraf "Data Hiding By LSB Substitution using Genetic Optimal Key-Permutation" International Arab Journal of e-Technology. Vol 2, No. 1, January 2.