



ISSN:1991-8178

Australian Journal of Basic and Applied Sciences

Journal home page: www.ajbasweb.com



Data Hiding in Multiple Frames Using Base64 Encoding

¹M. Pachhaïmmal@Priya, ²I.T. Sivaranjani, ³P. Saranya

¹Dept. of Computer Science and Engineering/Sri Sai Ram Institute of Technology, Chennai, India.

²Dept. of Computer Science and Engineering /Sri Sai Ram Institute of Technology, Chennai, India

³Dept. of Computer Science and Engineering /Sri Sai Ram Institute of Technology, Chennai, India

ARTICLE INFO

Article history:

Received 28 January 2015

Accepted 25 February 2015

Available online 6 March 2015

Keywords:

Frame Selection, Data hiding, Data Extraction.

ABSTRACT

Data hiding techniques can be used to embed a secret message into digital video streams for copyright protection, access control and transaction tracking. It can be used in military and defense to maintain security and privacy. Data hiding is used to conserve the confidentiality of the secret data at the same time maintaining the original video content. Some data hiding techniques are used to assess the quality of video in the absence of the original reference. The quality is estimated based on computing the degradations of the extracted hidden message. In this paper, Base 64 Encoding scheme, which is an asymmetric encryption algorithm, is used for enhanced security. Data hiding is also used for data concealment in applications of video transmission.

© 2015 AENSI Publisher All rights reserved.

To Cite This Article: M. Pachhaïmmal@Priya, I.T. Sivaranjani, P. Saranya., Data Hiding in Multiple Frames Using Base64 Encoding. *Aust. J. Basic & Appl. Sci.*, 9(10): 10-14, 2015

INTRODUCTION

Data hiding techniques can be used to hide the data into another efficiently, such that only the destined receiver can attain the hidden and original data. Existing systems uses Codeword Substitution, Motion vector difference, Discrete Cosine Transform and Intra Prediction Mode Encryption techniques to encrypt the video content. Techniques had been proposed to hide data into the video streams without knowing the original content. In addition, many works had been made using Paillier cryptosystem. However, due to the drawbacks of the Paillier cryptosystem, the encrypted images are of higher computational complexity and consumes much space than the original image. Data hiding is implemented

Using H.264/AVC reference software version JM-12.2. In the proposed system, a novel scheme for hiding data in video streams is presented, in which text data is hidden into image and the image is hidden into the video file. The secret image is split into two shares and is being hidden in any of the two random frames. The proposed system includes three parts, i.e., Frame selection, Data Hiding and Data Extraction. In Frame Selection module, the video is split and the frames in which the data has to be hidden is chosen. In Data hiding module, the encrypted text is hidden in image which in turn is split into two shares and embedded into two frames. In Data Extraction module, the video is decrypted

and the hidden text and image is being extracted without any degradation of the original video.

Frame Selection:

In this module, we select two frames for hiding the data. As a first step, we have to select a video in which data is to be hidden. After selecting the video, we have to split the video into three formats-audio, video and frames. We use ffmpeg tool for splitting the video. Ffmpeg is a tool which is generally used to extract audio from a video file, i.e., it converts an video file such as avi, mpeg, flv into an audio file such as mp3. It can be done by using the following command from command line prompt.

```
ffmpeg -i video.flv -ab 160k -ac 2 -ar 44100 -vn audio.mp3.
```

where

i → input

ab → bitrate(160kb/sec)

vn → no video output

ac 2 → 2 channels

ar 44100 → sampling frequency.

The ffmpeg tool is interfaced with the program so that it will be automatically called when the video is to be splitted. The ffmpeg tool will run in the command line prompt whenever it is called. After extracting the audio file from the video, the video is splitted into n number of frames. From these collection of frames, any two frames can be selected for hiding the data.

The converted audio file, video file and frames will be automatically generated and stored in the same folder as that of the source code. From these

automatically generated files, we can choose any two random frames for hiding the data.



Fig. 1: UI which is interfaced with ffmpeg tool for splitting the video.

Data Hiding:

In this module, we have to select an image and a data to be hidden into the video. In the existing data hiding projects, either data or an image can only be hidden. In our proposed project, we hide both the data and the image. We hide data into an image and then the image(with data) is hidden into the video. For this process, we first have to select an secret image to be hidden and embed the data. Actually, data cannot be hidden into an rgb image. So we should convert the RGB image into a grayscale image. It can be done by converting the RGB image into a grayscale image and then the grayscale image is converted into a binary image.

Images containing only intensity are called grayscale images. The values of the grayscale images range from [0,1]. An RGB image can be converted into a grayscale image by using the following command.

```
rgb=rand(200,200,3);
```

```
Gray=0.2989*rgb(:,:,1) + 0.5870 *rgb(:,:,2)
+0.1140*rgb(:,:,3);
```

Now, the converted grayscale image should be converted to a binary image. Only then data can be embedded into the image. Gray scale image can be transformed into a binary image by using the following pseudocode.

```
sz=size(Image);
mybin=zeros(size(Image));
for i=1 : sz(1)
    for j=1 : sz(2)
        if(Image(I,j)> Tvalue)
            mybin(i,j)=1;
        end
    end
end
```

The equivalent one line code is
Mybin(find(Image>Tvalue))=1;

Where

Tvalue → threshold value.



Fig. 2: UI which is interfaced for public key sharing.

The above for loop checks whether the value at each pixel position is greater than the threshold value. If the condition is true, then the value at that pixel position is set to 1. Otherwise, the value at that pixel position is set to 0. When this condition is checked for all the pixel positions of the image and the corresponding value is set, the resulting image is a binary image.

This binary image is splitted into two shares by using visual cryptography technique. The image which is splitted into two shares will remain just as a noise until the shares are overlapped. Now the secret data should be hidden into the image. The secret data is first converted into a cipher text using Paillier cryptography technique. This cipher text is then embedded into any one of the two shares by using

steganography technique which is shown in Fig4. Now the two shares are hidden into two randomly selected frames by using invisible watermarking.

Now, the frames are to be converted into video and mixed up with the audio using ffmpeg tool.

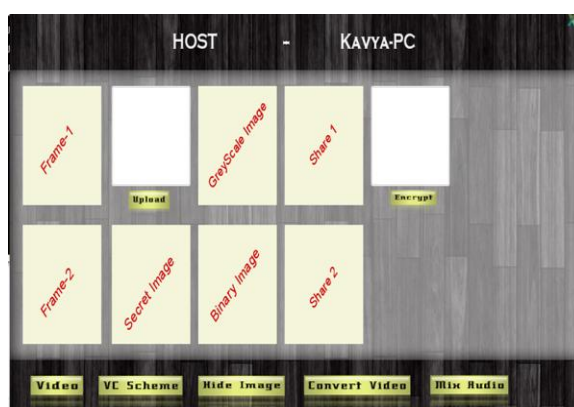


Fig. 3: UI which is interfaced for data embedding.

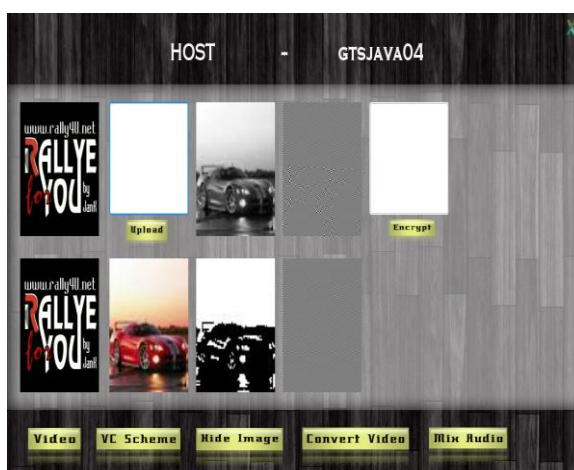


Fig. 4: UI after data embedding.

Data Extraction:

In third module, encryption is done by using destination's public key and the video is transmitted from the sender's end. In receiver's system, video is decrypted using his private key which is shown in Fig5 and split into three formats, i.e., video, audio and frames using ffmpeg tool. The corresponding Video file, Audio File and Frame file are generated. The frames which contain the secret data is identified by the intended receiver from the decryption key. The data is extracted from the frames which are watermarked. The shares are first extracted from the two frames. From one of the shares which contains the secret text, the text is extracted and decrypted. Then the two shares are merged together to get the secret binary image.

After the extraction, the image received will be noisy, so it is reconstructed to get the original image which is shown in Fig6. The original video can be received without any quality loss using simulated algorithm. The decryption is done using Base 64

scheme, which is an asymmetric algorithm. Hence, security and privacy can be achieved in a greater way, since the decryption key cannot be derived from the public encryption key using asymmetric algorithm. Experimental results have shown that asymmetric key algorithms are more efficient on preserving the integrity and confidentiality of the information.

The processed and output images such as the frame 1, frame 2, share 1, share 2, grayscale image, binary image and the original secret image are stored in a separate file as output. The secret text is stored separately in a file after decryption. The video is finally merged using the ffmpeg tool and the original video file is obtained without any content loss or quality degradation.

Conclusion:

In this paper, a novel model is presented to embed additional data and secret image into video streams which contains frame selection, data hiding

and data extraction phases. The main objective of this paper is to overcome the shortcomings of the previous hiding and encryption algorithms. This paper presents a model of hiding the image into two random frames of the video stream. It is impossible for the intruder to find the secret image, even if one of the frame which contains one of the share of the

image is found. The algorithm can preserve the bit-rate exactly even after encryption and data hiding. It is also simple to implement. Experimental results on existing systems have shown that the encryption and data hiding scheme can conserve file-size, whereas the degradation in video quality exists.

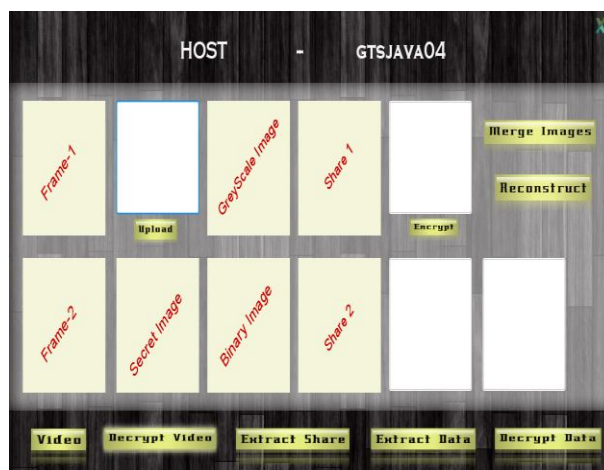


Fig. 5: UI interfaced for decryption and data extraction at receiver side.

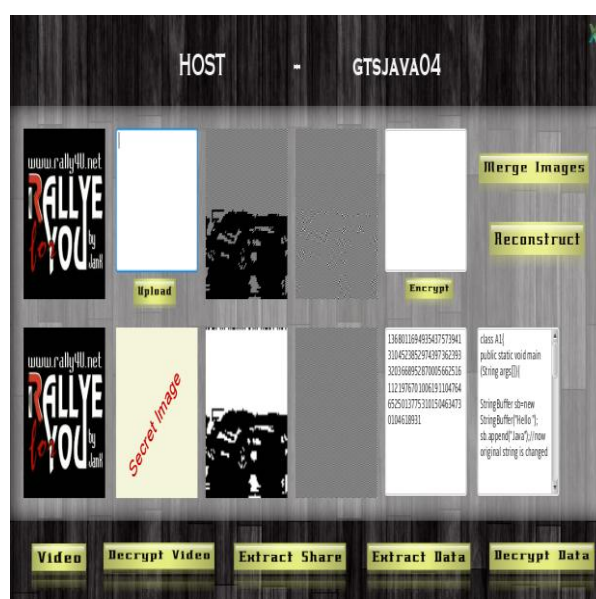


Fig. 6: UI after decryption and data extraction at receiver side.

REFERENCES

- Hong, W., T.S. Chen and H.Y. Wu, 2012. "An improved reversible data hiding in encrypted images using side match," IEEE Signal Process. Lett., 19(4): 199-202.
- Lu, W.J., A. Varna and M. Wu, 2011. "Secure video processing: Problems and challenges," in Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing, Prague, Czech Republic, 5856-5859.
- Ma, K.D., W.M. Zhang, X.F. Zhao, N. Yu and F. Li, 2013. "Reversible data hiding in encrypted

images by reserving room before encryption," IEEE Trans. Inf. Forensics Security, 8(3): 553-562.

Puech, W., M. Chaumont and O. Strauss, 2008. "A reversible data hiding method for encrypted images," Proc. SPIE, vol. 6819, pp. 6819E-1-6819E-9, Jan.

Subramanyam, A.V., S. Emmanuel and M.S. Kankanhalli, 2012. "Robust watermarking of compressed and encrypted JPEG2000 images," IEEE Trans. Multimedia, 14(3): 703-716.

Zhang, X.P., 2012. "Separable reversible data hiding in encrypted image," IEEE Trans. Inf. Forensics Security, 7(2): 826-832.

Zhang, X.P., 2011. "Reversible data hiding in encrypted image," IEEE Signal Process. Lett., 18(4): 255-258.

Zhao, B., W.D. Kou and H. Li, 2010. "Effective watermarking scheme in the encrypted domain for buyer-seller watermarking protocol," Inf. Sci., 180(23): 4672-4684.

Zheng, P.J. and J.W. Huang, 2012. "Walsh-Hadamard transform in the homomorphic encrypted domain and its application in image watermarking," in Proc. 14th Inf. Hiding Conf., Berkeley, CA, USA, pp: 1-15.