IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 1, Ver. 5 (Jan. 2014), PP 71-82
www.iosrjournals.org

Reversible Encrypted Data Hiding In Encrypted Video

Biby Bino Varghese[1] and Rosna P. Haroon[2]
[1,2]Department of Computer Science and Engineering, Ilahia College of Engineering and Technology, Kerala,

Abstract: *A data hiding scheme for encrypted image has been proposed by Xinpeng Zhang [11]. In this paper however, we put forward a data hiding scheme on an encrypted video. The data to be embedded is encrypted ensuring greater security. Each video frame is a jpg image, which is then encrypted using any stream cipher. The data to be embedded is encrypted. It is then embedded by modifying a small portion of the encrypted cover image. The decryption of video is done first followed by the decryption of the data. With the aid of the data hiding key and spatial correlation in natural image, the embedded data can be successfully extracted and the original video can be perfectly recovered. The accuracy of receiver is very much dependant on the block size of the cover image and consequently, an optimum block size is suggested as an experimental value.*
Index Terms: *Video Encryption, Data Embedding, Frames*

I. Introduction

Data embedding helps data to be stored inside any cover media. The cover media may be image, audio or video. Reversible data hiding ensures that the cover media is unaltered after decryption and data extraction. That is, the cover that was used for data embedding shoud be as such recovered after data extraction.

Encryption is a technique that ensures secure sharing of data. Only the sender and receiver would have access to the encrypted data, by means of the key. Encryption may be symmetric or asymmetric, based on the keys being used. Joining the two schemes of data hiding and encryption provides added security and convenience in transmission.

Various techniques have been proposed to hide data in images since early times [1]. As known, a video is a sequence of images also called as frames. Hence, the techniques that can be applied in embedding data into an image can also be applied to video streams by applying them to each individual frames. These methods include least significant bit-modifications, masking and filtering, transformations [2], transform domain embedding and real time video steganography [3].

Data hiding in video is much way similar to digital watermarking in video. The hidden data must be transparent to a Human Visual System (VHS) and must be imperceptible. It is desirable that the embedding/extraction methods are resistant to attacks. Consecutive frames of a video may look alike. In such cases, new frames that look similar can be inserted into the stream, or new frames can replace the existing frames [4]. Because each frame carries some hidden data, videos can have an enormous amount of data embedded.

In the method of data hiding in single images only less amount of data can be embedded. Whereas by varying the medium to a video, a stream of continuous data can be embedded. The proposed method is used to hide data in a video sequence. Initially the video that acts as the cover media is encrypted. The method then accepts the data that has to be embedded from the user. Post encryption of the data entered, it becomes capable of being embedded into frames, each of which is encrypted. At the receiver side, the data is retrieved and decrypted. The cover media is recovered after data extraction. Since the method is reversible, there must be as little distortions.

II. Related Works

The author Esen E. Alatan discusses about a powerful block based method that uses forbidden zone data hiding. This method allows no alteration in data hiding process in the host signal range. It is resilient to attacks including frame manipulation. In this method, the video is divided into frames. The frame is then partitioned into a number of zones. Some of the zones are forbidden from data hiding and the others are chosen for embedding. This paper is very much similar to the present paper in terms of frame construction from video. Both papers also segment the image into zones and blocks.

Just Another Watermarking System (JAWS) [6] developed by Philips Research is a good performance watermarking system used for real time and DVD applications. A reference key is generated in this method, which then is used to generate a watermark pattern, shaped for individual frames. Though this is a tedious method owing to the number of reference keys, it provides a higher degree of security.

Segmenting the frames into blocks before data embedding is not a new concept. The H.264 video encoder uses various block sizes during inter prediction as opposed to a fixed size used in the present paper. Using this property of varying block sizes, desirable data can be hidden and extracted without knowledge of the original video content [7]. Based on the size of the block or the block type in the frame, a binary number is assigned to each of the blocks. The message to be embedded is also converted to binary bits and two of them are paired. This method then hides the bit pair in blocks, either consecutively or in a predefined pattern.

By embedding data in video, we emphasize on being able to hide more data in the video than in an image. The rate of data to be embedded can also be varied dynamically based on the type and quality of frame in the video and the reference quantization parameter [8]. Here an adaptive data hiding scheme with a varying embedding rate gave emphasis to the type of frame and on the MPEG-2 reference quantization parameter, which depends on the spatial activity in the macro blocks of a frame.

As there are a variety of parameters at which the encoder may hide the data, the different parameter sets to be used need to be pre-decided and made known to the decoder. However, at runtime, the parameter set per frame need not be explicitly sent to the decoder since the decoding generally becomes successful only if the decoder uses the right combination from its known set of parameters.

Blind data hiding schemes can be used when hiding fewer amounts of data. However in hiding large volumes of data, videos always have a profound edge when compared to images. This can be attributed to the fact that, a single video can be divided to multiple frames. Each frame is an image. Data hiding in video hides data in such multiple frames. Data hiding in image has only a single frame to act as the cover media.

The use of formulas in hiding data into the frame creates a higher level of complexity. Thus it strengthens the algorithm. Methods like Entropy Thresholding Scheme [9] or Selectively Embedding in Coefficients Scheme use various formulas in data hiding. These methods, applied to images can easily be extended to videos.

Entropy Thresholding Scheme divides a frame into a number of non-overlapping blocks. A discrete cosine transform of these blocks are taken. Energy of these blocks is then calculated based on some formulas. Blocks that have energy below the threshold energy are only used for data hiding. In Selective Coefficients method, the frame is partitioned into a number of zones. Some of the zones are forbidden from hiding any data into them while the others are chosen for embedding.

Most data hiding schemes use uncompressed video data. Compression of the video may bring forth certain errors in the stream, which are marked using repeat accumulate (RA) codes. A block – based selective embedding type data hiding encapsulates forbidden zone data hiding (FZDH) [5] and RA codes [10] with an additional temporal synchronization mechanism. Here, each block is partitioned into two groups. One group is used for frame marker embedding and the other is used for message bits. By means of simple rules applied to the frame markers, certain level of robustness against frame drop, repeat and insert attacks are brought in. This method is used for both MPEG-2 and H.264 videos.

III. PROBLEM DOMAIN

The rapid growth in the demand and consumption of the digital multimedia content in the past decade has led to some valid concerns over issues such as content security, authenticity, and digital rights management. Multimedia data hiding, defined as imperceptible embedding of information into a multimedia host, provides potential solutions, but with many technological challenges.

Ever improving network bandwidths, computer speeds, digital storage capacities, and wireless capabilities are changing our lives right from the way we entertain ourselves, communicate with each other, or assimilate and disseminate knowledge, to the way we operate our bank accounts. A key driver for these changes has been the rapid growth in the demand and consumption of digital multimedia content. This has, however, lead to some valid concerns over multimedia content security, authenticity, and intellectual property rights. Because of its potential applications in multimedia content security, data hiding or data embedding continues to receive considerable attention from the research community.

The general framework of a data embedding system is shown in Fig. 1.

Data hiding or steganography is the art and science of communicating in such a way that the very existence of communication is not known to the third party. It is very important to investigate steganalysis, the study of techniques to detect the presence of hidden data. Also significant is to understand the limits of steganography, and analyze how much information can be embedded into images, audio, or video hosts without being detected.

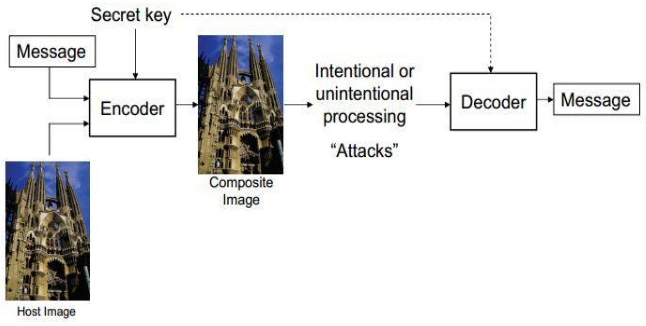

Figure 1. General framework of a data hiding system [12]

Data embedding can be done onto images, videos, and audios imperceptibly to human beings. These methods can be used for covert communications, watermarkings, copyright protections etc.

Steganalysis is the art and science to detect whether a given medium has hidden message in it. Data can be embedded in video, audio or text files. Data embedding in video and images are called watermarking, steganography etc. Data hiding in audio is called audio watermarking and is widely used for copyright protection.

Attacks on the data hiding methods also known as steganalysis have now become a widespread threat. It poses a big challenge to overcome these attacks. Security for embedded data on multimedia can be achieved to a large extent through encryption.

Encryption is the process of encoding messages (or information) in such a way that eavesdroppers or hackers cannot read it, but that authorized parties can. In an encryption scheme, the message or information (referred to as plaintext) is encrypted using an encryption algorithm, turning it into an unreadable cipher text (ibid.). This is usually done with the use of an encryption key, which specifies how the message is to be encoded. Any adversary that can see the cipher text should not be able to determine anything about the original message. An authorized party, however, is able to decode the cipher text using a decryption algorithm that usually requires a secret decryption key, which adversaries do not have access to.

There are two basic types of encryption schemes: Symmetric - key and public - key encryption. In symmetric - key schemes, the encryption and decryption keys are the same. Thus communicating parties must agree on a secret key before they wish to communicate. In public - key schemes, the encryption key is published for anyone to use and encrypt messages. However, only the receiving party has access to the decryption key and is capable of reading the encrypted messages. The importance of multimedia data and the importance and volume of the data to be embedded increases the importance of security on multimedia transactions with embedded data.

Hence, we may go for data embedding also as an additional protection. Data embedding or data hiding can be done on audio, video and textual files. Data embedding is done in a multimedia file owing to the data's confidential value. It thus is important to keep the embedded data secure. Also, it must not be extracted by any third party attacks. The security of the system is compromised when a third party gains access to the data.

IV. PROBLEM IDENTIFICATION

There are three main conflicting requirements of a multimedia data embedding system: perceptual transparency, robustness, and capacity. Information embedding into a multimedia host should not incur any perceptual distortion to the host, i.e., the composite signal should be perceptually transparent. The data should be recoverable even after the composite multimedia signal has undergone a variety of attacks, intentional or unintentional, to remove the embedded data. In other words, the hidden data must be robust against a variety of attacks. We would also like to embed as many bits into the host as possible, or, the capacity of the embedding system should be high.

Data embedding or data hiding can be done on audio, video and textual files. Data embedding is done in a multimedia file owing to the data's confidential value. It thus is important to keep the embedded data secure. Also, it must not be extracted by any third party attacks. The security of the system is compromised when a third party gains access to the data.

Images are a more common cover media for data embedment. However, videos pose as a better cover owing to two reasons. One of the reasons is that more data can be embedded into a video as compared to an image. Videos have as many images in them as there is the number of frames. Each of these frames becomes capable of holding data in them. Thus, the embedding capacity of a video is way greater than an image. Data can be embedded into video in any particular order. This order can be kept hidden and shared only between the sender and receiver. Such methods can add to the security.

V. PROBLEM STATEMENT

Consider a military scenario, where a captain A wants to send terrestrial reports to another captain. Because of confidentiality reasons, he encrypts the image and passes the document to his assistant B, so that the mail may be forwarded. Now B is required to add some more description. Since he has no authority to send it as a separate file, he embeds it to the report send to him from A. Then he sends the complete file to the destination. The procedure follows as follows:

a. Encrypt the image.
b. Embed the data

There is no security to the data being embedded in this manner.

In this paper, the data that is being embedded is encrypted before embedding. Also, when A sends a video file, an efficient hiding scheme needs to be proposed. Larger amounts of data can be hidden to the video file by B.

In the proposed paper, embedding data to the video file proceeds as follows:

a. Divide the video to frames.
b. Encrypt each frame.
c. Encrypt the data to be embedded.
d. Embed the data

VI. PROPOSED SYSTEM

This section describes the proposed scheme, the phases of data hiding in video and their algorithms. Since a video is created from a sequence of still images called frames, the same technique of image data hiding can be extended to the concept of video data hiding.

The phases of data hiding that has to be passed through for an effective embedding in video are stated below.

- Frame Construction
- Frame Encryption [11]
- Data Encryption
- Data Embedding [11]
- Frame Decryption [11]
- Data Extraction and frame recovery [11]
- Data decryption
- Video Construction

The cover media, here the video, is initially divided into multiple frames. Each of these frames is an image. The frame is encrypted using any one of the convenient encryption mechanisms with a frame key. The frame key is symmetric. That it, the sender and receiver uses the same key in encryption and decryption. After the data to be embedded is accepted, it is encrypted using a data key. The encrypted data in binary is embedded into the frame using algorithms specified in [11]. At the receiver side, initially the frame is decrypted with the frame key and the data is extracted using algorithms in [11]. This data is then decrypted using the same data key.

The proposed system can be represented as given in Fig.2.

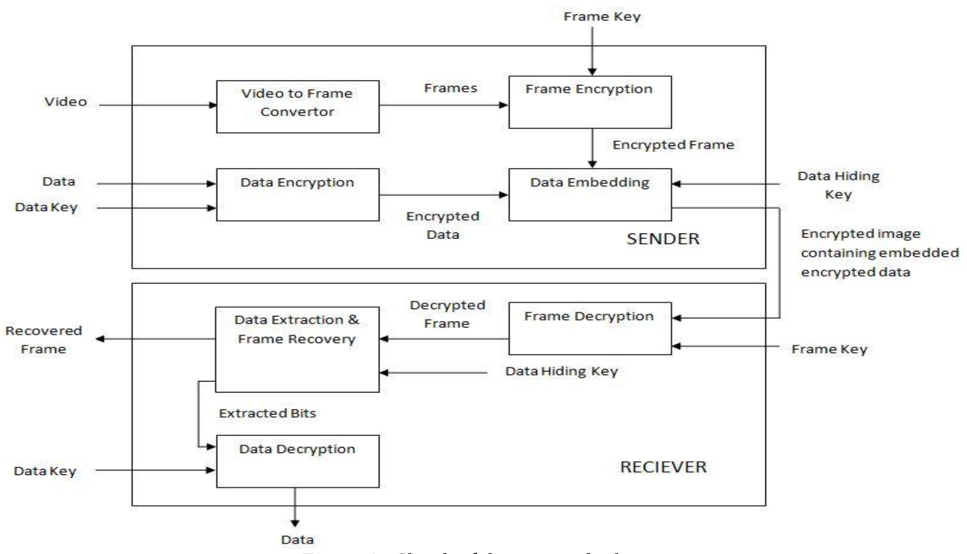

Figure. 2. Sketch of the proposed scheme

I. Frame Construction

In [11], a single image is used to hold the data. Only fewer amounts of data can be embedded into an image media as compared to a video cover media. Whereby varying the medium of [11] to a video, a stream of continuous data can be embedded.

In this paper, initially, the video that intends to acts as the cover is input from the sender. It is converted to an image sequence, with a fixed frame rate, both at the sender and receiver side. Various algorithms and software can be used to convert a video into frames. These will result into a sequence of frames, and we can apply the data embedding algorithms on each frame.

The frame construction process can be represented schematically as in Fig. 3 This phase begins by reading in an input video which is to be segmented to frames. The total duration of the video is calculated and stored to a constant. Snapshots of the video stream are taken at each time interval *i*, where *i* is dependent on a previously fixed frame rate. The frame rate is fixed and is same at the sender and receiver end. Each of these snapshots is given a unique name. These unique names help in distinguishing between consecutive frames. They also assist when the frames are to be joined back into a video stream. After naming the snapshots, now called frames, they are stored into the database in an ordered manner.

The process iterates till the end of the video duration. When *i* is any greater than *video_length*, the total video duration, the iteration is stopped.

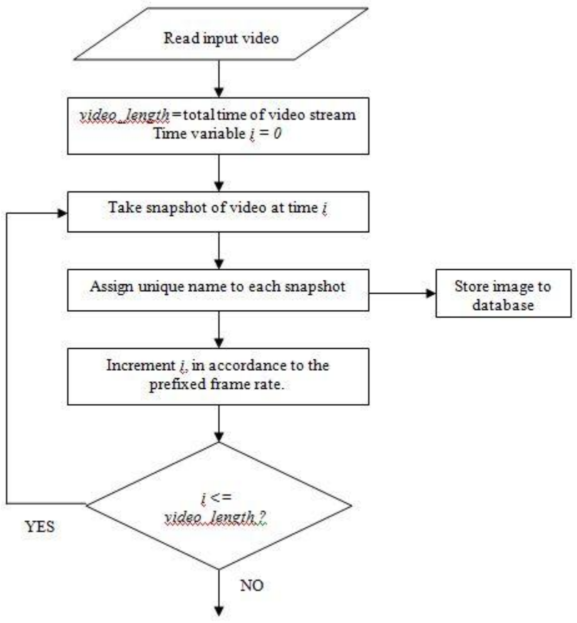
Read input video
video_length=total time of video stream
Time variable i = 0
Take snapshot of video at time i
Assign unique name to each snapshot
Store image to database
Increment i, in accordance to the prefixed frame rate.
i <= video_length ?
YES
NO

Figure 3. Frame Construction from a video stream

The algorithm of Fig.3 can be implemented through any of the softwares like DVDVideoSoft, FormatFactory, XVideo Converter. Most of these softwares support video files of any type. Apart from using softwares, codes can be created for the same in Java or MATLAB.

To ensure reversibility, [11] embeds data on gray scale images, so that the cover media will be recovered as such, on the receiver side. However, every frame of the image sequence generated using the algorithm in Fig.3 may not be a gray scale image. The scheme proposed do not process color images, so it becomes necessary to convert these into pixel values that confine to [0,255]. Pixels in this range are always grayscale. The RGB frames are converted to grayscale using an RGB to grayscale converter, illustrated diagrammatically in Fig.4.

The converter works using an averaging function. Every pixel in an image is made up of its own red, green and blue (RGB) components. When a frame from the image sequence is passed into a RGB to grayscale converter, it scans each of the pixels from top left of the image to its bottom right.

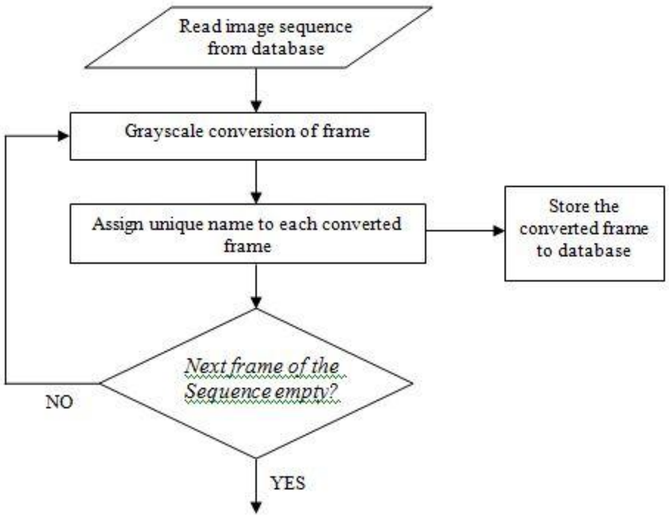
Read image sequence from database
Grayscale conversion of frame
Assign unique name to each converted frame
Store the converted frame to database
Next frame of the Sequence empty?
NO
YES

Figure 4. RGB to Grayscale Conversion of frames

An average of these three components is found by adding up the values of the three components and dividing them by three. The resulting value lies in the range [0,255]. This converter functions as by (1).

$$pix\_val = (comp\_r + comp\_g + comp\_b) / 3 \quad (1)$$

where *comp_r*, *comp_g*, *comp_b* are the red, blue and green components and *pix_val* is the new pixel value after grayscale conversion.

In this paper, the audio component of the video, thus of the frames, is not taken into account. Embedding data into audio is called audio watermarking and is used for copyright protection purposes.

II. Frame Encryption

The processing images used in this phase are each of the individual frames. These frames are of grayscale values [0,255]. The frame construction phase converts them to grayscale from RGB colour frame.

In [11], every bit of every pixel is modified before frame encryption. However, in the proposed frame encryption, the bits are not modified. Once the frames are constructed and converted to grayscale, each frame is straightaway encrypted.

One simple way to encrypt and decrypt data is using XOR Encryption method. The character in the data stream and those of a code key are XORed together to create and encrypted character. The process is exactly same for encryption as well as decryption. The code key can be made up of any alphanumeric characters you want. It can be any long, much longer key used it is most tough to crack.

Vernam or XOR Ciphers turn out to be easy to break. Despite its weakness in simple applications, the XOR Encryption remains an important cipher. It is weak when we use repeating keys, but it can be very effective when the key stream varies continuously. This approaches is often uses in modern stream ciphers RC4 (Rivest Cipher #4), developed by Ron Rivest of RSA data security.

For frame encryption, XOR encryption is used. A random sequence of bits acts as a key for this encryption method. As in [11], the random bits are also known as the frame key. XOR encryption uses symmetric keys. That is, the sender and receiver use the same key in encryption and decryption. Frame keys are used in encrypting the bits of each pixels of the image to create an encrypted image. Each bit of the frame key is XORed with each bit of the pixel to create a totally new pixel vaue.

As Fig 5. depicts, there are two inputs to the frame encryption phase. They are the frame key and a single frame from the database. Every pixel of the frame is scanned in order. All the eight bits of every pixel are XORed with the frame key.

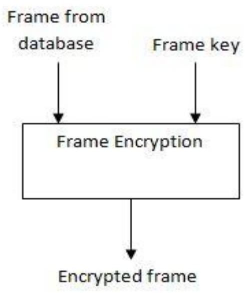

Figure 5. Frame Encryption

Any of the stream cipher techniques can be used for frame encryption. Further, these techniques can also be coupled with the XOR method. This would provide more layers of security because a potential attacker would have to break through all the encryption techniques employed in frame encryption. A number of secure stream cipher methods can be used here to ensure that anyone without the encryption key, that is the frame key, such as a potential attacker or the data hider, cannot obtain any information about original content from the encrypted data.

III. Data Encryption

Once the cover is encrypted, the data that is to be embedded into it has to be made ready. The data that needs secure transmission is accepted from the user. This data is usually in textual format. Encryption mechanisms require it to be converted to a binary sequence.

Converting text to Binary is a two step process. First each letter or character or number is converted to its decimal equivalent using an ASCII (American Standard Code for Information Interchange) chart. The second step creates a binary equivalent for the decimal. These binary equivalents for each decimal are entered into a string and the string is named b_i.

A data key d_i is also entered from the user. This key is symmetric. That is, the sender and receiver use the same key for encryption and decryption. This data key encrypts the binary sequence to create an encrypted data. This encrypted data is then ready to be embedded into the cover frame.

The encryption is done by the principle of XOR cipher. In cryptography, the simple XOR cipher is a type of additive cipher, an encryption algorithm that operates according to the principles:

$$A \oplus 0 = A$$
$$A \oplus A = 0$$
$$(A \oplus B) \oplus C = A \oplus (B \oplus C)$$
$$(B \oplus A) \oplus A = B \oplus 0 = B$$

where $\oplus$ denotes the exclusive disjunction (XOR) operation.

With this logic, a string of text can be encrypted by applying the bitwise XOR operator to every character using a given key. To decrypt the output, merely reapplying the XOR function with the key will remove the cipher.

Fig.6 illustrates a typical mechanism used for data encryption.

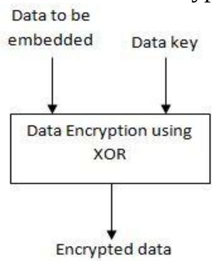

Figure 6. Data Encryption

The binary sequence b_i is encrypted by XORing it with a data key d_i. Thus an encrypted stream B_i is obtained as in (2).

$$B_i = b_i \wedge d_i \qquad (2)$$

B_i is the encrypted value of the data to be embedded. It is a binary string. When the length of b_i, the data to be embedded exceeds the length of d_i ; d_i is repeated until it reaches the length of b_i. The encrypted output stream B_i is then sent for being embedded into the frame. Since *Bi* is encrypted with a key known only to the sender and receiver alone, it is more secure.

The data encryption phase too uses the XOR encryption technique. The encryption method XOR is also called as additive cipher. It is a strong and unbreakable encryption technique provided the key is as long as the plain text, the key is completely random and is never used again between two communicating parties. It is simple to implement with no tradeoffs on security aspects.

IV. Data Embedding

With an encrypted frame and encrypted data, although a data-hider does not know the original image content, he can embed additional message into the image by modifying a small proportion of encrypted data.

As stated from [11], initially the data-hider segments the encrypted image into a number of non-overlapping blocks sized as sxs. Each block in the frame can carry one bit each. Thus, a frame contains as much hidden data as the number of blocks in it. There are s^2 pixels in each block.

Two sets S0 & S1 are created based on the data hiding key and pixels are assigned to each of it. Here, the chances that a pixel belongs to S0 or S1 is 0.5.

If the additional bit to be embedded is 0, flip the 3 least significant bits (LSB) of each encrypted pixel in S0. If the additional bit is 1, flip the 3 encrypted LSB of pixels in S1. The other encrypted data are not changed. This phase is illustrated with Fig.7.

This process is repeated until every frame of the video expires and there is no more space for accommodation of data. Else, the process continues until the data to be embedded is completely hidden in the cover.

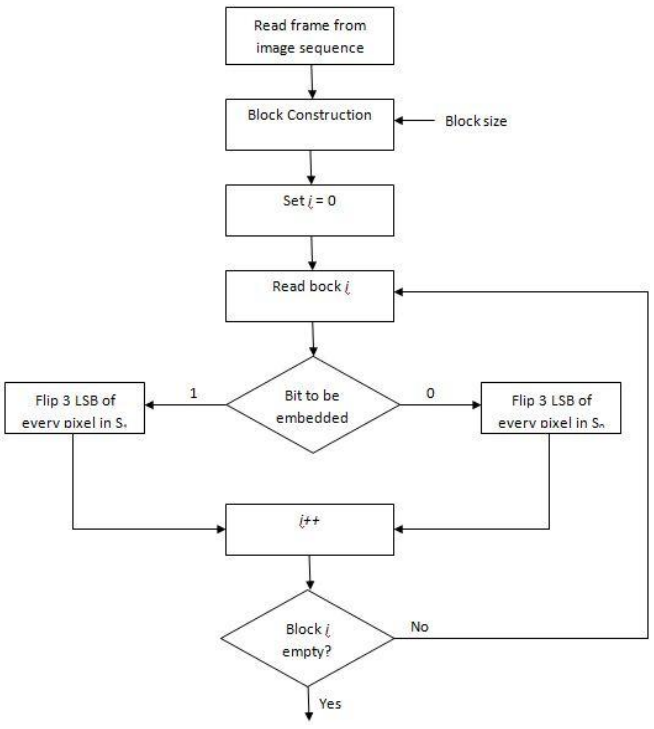
Read frame from image sequence
Block Construction
Block size
Set i = 0
Read bock i
Bit to be embedded
1
0
Flip 3 LSB of every pixel in S1
Flip 3 LSB of every pixel in S0
i++
Block i empty?
No
Yes

Figure 7. Data Embedding

Every block in the frame are all of the same size. Each block in this method is capable of holding a single bit. If the size of the blocks in the frame is increased, the number of blocks in the frame decreases. This in turn decreases the number of bits that can be embedded in a single frame. Ultimately the embedding capacity of the video is decreased drastically.

V. Frame Decryption

Decryption of the frame is done similar to its encryption and proceeds as stated in [11]. The receiver receives an encrypted frame containing embedded data. No data can be read from the obtained file at this stage as it is encrypted. This particular attribute keeps the file secure from third party attacks. Unless an attacker has the key to decrypt the frame, the data is protected. Thus, transmission between sender and receiver is secure.

A key, named frame key is initially shared between the sender and receiver. The encrypted image containing embedded encrypted data has to be decrypted using this frame key. The obtained frame and the shared secret key are XORed. The resulting pixel values form a new frame that accounts to be the original frame.

After decryption, the original image that was sent from the sender is obtained. This image contains the embedded data.

VI. Data Extraction & Frame Recovery

Data is extracted and frame is recovered in the same manner as in [11]. Having obtained the decrypted cover image, the embedded data can now be retrieved from the same. Initially, the receiver segments the decrypted frame into various blocks. The sender and receiver are both aware of the size of these blocks. It is always fixed at a constant value. The pixels in each block are divided into two groups based on the data hiding key, as it was done in the data embedding phase. The phase begins by scanning pixels one by one from the top left of the frame.

For each pixel, if the embedded bit in the block including the pixel is zero and the pixel belongs to S_1, or the embedded bit is 1 and the pixel belongs to S_0, the data-hiding does not affect any encrypted bits of the pixel. In the embedding process, only three of the least significant bits were used. That is, the other five bits or the five MSB of all pixels should remain intact after decryption as data embedding would not modify them.

On analyzing each block, the receiver flips all the three LSB of pixels in S_0 to form a new block, and flips all the three LSB of pixels in S_1 to form another new block, named as H_0 and H_1 respectively.

Blocks H_0 and H_1 are created by flipping least significant bits of S_0 and S_1. Either H_0 or H_1 is the original block from the sender side. The original block can be figured out by measuring the correlation between the pixels. The original block has a higher correlation between pixels. That is, any particular pixel and its adjoining pixels are spatially correlated. A fluctuation function f_0 and f_1 is calculated for both sets H_0 and H_1 to find the original block as in (3) [11].

$$f = \sum_{u=2}^{s-1}\sum_{v=2}^{s-1}\left| p_{u,v} - \frac{p_{u-1,v} + p_{u,v-1} + p_{u+1,v} + p_{u,v+1}}{4} \right| \quad (3)$$

The set with the lower value of fluctuation function is selected to be the original block. If $f_0<f_1$, regard H_0 as the original content of the block and let the extracted bit be 0. Otherwise, regard H_1 as the original content of this block and extract a bit 1. Finally, concatenate the extracted bits to retrieve the embedded encrypted message and collect the recovered blocks to form the original frame. This block is then appended to help complete forming the actual frame.

VII. Data Decryption

The decryption of data uses XOR mechanism as it is the same mechanism used at the sender side. Using the data key and the XOR technique, the data collected at the receiver side is decrypted. The data decryption phase is represented in Fig.8.

As a part of data extraction, a string of binary data is extracted from the cover frame. However, this is an encrypted stream of binary data. The data to be embedded into the video was encrypted before hiding. Hence it is necessary to decrypt the data to obtain the actual data.

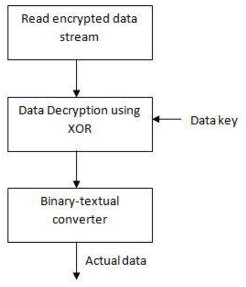

Figure 8. Data Decryption

A data key is input from the receiver that is meant to decrypt the data. The data key is known only to the sender and receiver. It is a symmetric key, in that the sender and receiver use the same data key in the encryption and decryption of data.

B_i is the sequence of bits extracted from the frame. It is a binary stream that is encrypted. It is decrypted using (3).

$$b'_i = B_i \wedge d_i \qquad (3)$$

B_i is XORed with the data key d_i to create a new decrypted sequence of bits *b'*. *b'* is the actual data embedded. *b'* is also in binary format. It has to be converted back into textual form. This occurs in two steps. Initially, ASCII codes are generated from *b'*. As the next step, textual equivalents of these codes are figured from the chart. Thus the actual data is obtained.

An additional property of this scheme is that, a person can never obtain or decrypt the embedded data until he successfully decrypts the frame. That is, to obtain the data embedded, a person has to have both the frame key and the data key. A person who has the frame key alone, but no data key, may decrypt the frame. But he has no access to the data, as it is encrypted. A person, even though he has the data key, but no frame key, can never access the data. This is because he has not decrypted the cover frame.

VIII. *Video Construction*

This is the last phase of the proposed scheme. Hidden data is extracted successfully from all the frames. It is decrypted to obtain the actual data. Also, all of the frames are perfectly recovered. Since these frames were given unique names, they can now be concatenated to form the original video that was send from the sender side.

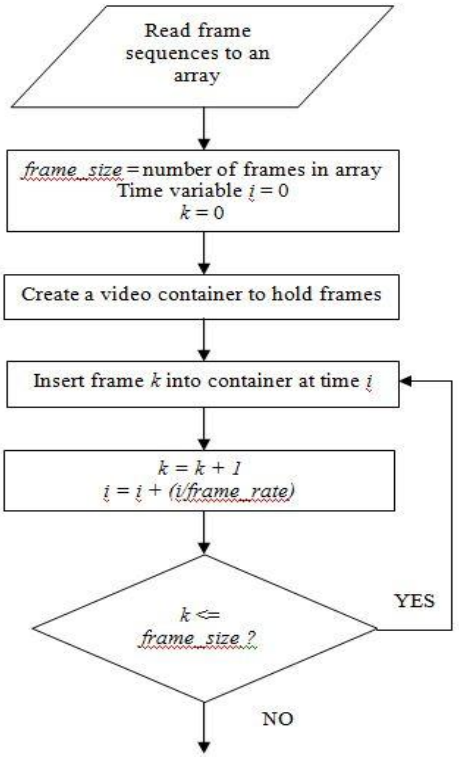
Read frame sequences to an array
frame_size = number of frames in array
Time variable *i* = 0
k = 0
Create a video container to hold frames
Insert frame *k* into container at time *i*
k = *k* + *1*
i = *i* + *(1/frame_rate)*
k <= *frame_size ?*
YES
NO

Figure 9. Video Construction from frames

In the video construction phase, the frames collected from the data extraction and frame recovery phase are concatenated to form an image sequence. It consists of an array of frames, each that has a unique name, possibly identified by its position in the original video stream. This array of frames is then joined to form a video stream, keeping in mind the predefined frame rate. This phase can be represented diagrammatically as in Fig. 9.

Initially, sequences of frames are read in for video construction. This sequence or array is only a mode of storage for image frames. It cannot be played like an ordinary video. To satisfy the purpose of obtaining the actual video, these frames are joined as to form a single file. Each frame from the sequence is inserted to a video container at fixed intervals. The interval is decided by the frame rate.

Fig.9. depicts the frame rate as a value which is known both at the sender and receiver sides. The interval is incremented until all frames are bound to the video container. As a result, the container outputs a video file that can be played with any player.

VII. SIMULATION

Java is used as the programming language for implementing the framework. The Java.net package is useful for socket programming, thus providing the inter-communication between data owner, Client and the CSP. NetBeans is an Integrated Development Environment/Interactive Development Environment mainly aims to create an efficient software development environment providing the required tools. NetBeans IDE provides comprehensive support to most of the newest Java technologies. The functionalities of IDE are provided as modules.

The proposed work has been implemented in java. Java is a general purpose, concurrent, class-based, object oriented programming language that is intended as "write once, run anywhere". Java has many packages that helps in its successful running. The image processing parts of Java are buried within java.awt.image package.

The model has been simulated in a Linux environment under the Java taking aid of the Ffmpeg software. Ffmpeg is a multimedia framework able to decode, encode, transcode, mux, demux, stream, filter and play most formats. It is easy to use, fast, light and saves time. It can easily convert video to audio or images.

In Fig.10, the software sequence model of the proposed work is depicted. The video stream that acts as the cover media is sent to Ffmpeg software. Sequences of frames are created in this process. These frames are then input to the project, run in Java.

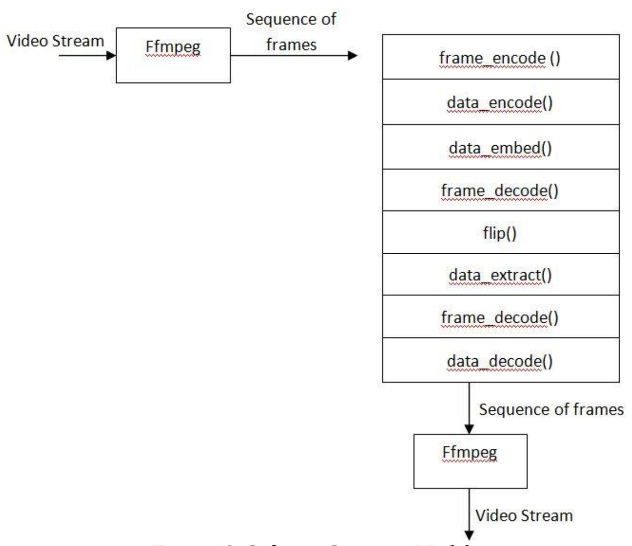

Figure 10. Software Sequence Model

Java code encodes each frame using the frame_encode() function. Data is encrypted using the data_encode(). Once the encrypted data and encrypted frames are ready, encrypted data is embedded into the encrypted frame using data_embed(). A flip() function is used in data embedding. This function flips the least significant bits of a pixel, depending on the bit to be embedded. The next function frame_decode() decodes the frame at the receiver end. data_extract() is used to extract data from the frame. Once data is extracted, the frame is recovered. The recovered frame is decrypted along with the extracted data using frame_decode() and data_decode().

VIII. RESULTS

Results are obtained and verified for embedding data in an encrypted video stream. The cover video is divided to frames. An encryption key obtained from the user encrypts the cover image. The data hiding key encrypts the data to be embedded.

The experimental results display a higher embedding capacity for a video, as compared to a single video frame, or an image. The block size of the proposed scheme is fixed with a side length of 32 [11]. With this block size, a single 512x512 image can have 16 blocks. With each block holding a single bit, such an image can hold 16 bits or two 8-bit characters. However, a video with 80 frames can hold 80 two bit characters, each frame being capable of holding two 8-bit characters.

Thus, the embedding capacity of a video is drastically higher than an image of the same frame size and block size. Under a fixed frame size of 512x512 and a fixed block size of 32x32, Table 2 compares the embedding capacity of an encrypted video and an encrypted image.

Apart from the increase of embedding capacity, this method also provides security. This is because unless the user provides the key to decrypt the cover media, he cannot extract the data from the video.

Table 2. Comparison study between encrypted video and image

	Encrypted Video	Encrypted Image
Frame size	512x512	512x512
Block size	32x32	32x32
Number of frames	50	1
Embedding capacity (in bits)	800	16

IX. Conclusion and Future work

Current communication systems impose a lot of significance to methods like data hiding and watermarking. In real life scenarios like medical or military, data hiding needs additional data to be securely embedded into any media by a data hider who has fewer priorities than the content owner. The proposed scheme can be used in areas where additional data has to be securely embedded by a data-hider who has fewer priorities than the content-owner. In this paper, encrypted data is embedded into an encrypted cover video. The video is divided into frames, with a fixed frame rate. Since the data to be embedded in the video is encrypted, it adds more security. The proposed scheme thus aims at enhancing security. Encryption in this paper is done with XOR, although other stream cipher methods can be adopted. At the receiver end, the video is perfectly recovered. The data bits are collected, from each block of every frame and joined to form a whole sequence and then decrypted to get the original data. Our paper manually provides the block size of the frames. As a future work, algorithms can be devised that automatically finds the optimum block size for embedding.

References

[1] W.Bender, D.Gruhl, N.Morimoto and A.Lu (1996). Techniques for Data Hiding, IBM Systems Journal. 35(3,4)

[2] Arup Kumar Bhaumik, Minkyu Choi, Roslin J. Robles and Maricel O. Balitanas (2009, June). Data Hiding in Video, International Journal of Database Theory and Applications. 2(2)

[3] A.K.Al-Frajat, H.A.Jalab, Z.M.Kasirun, A.A.Zaidan and B.B.Zaidan "Hiding Data in Video File: An Overview, Journal of Applied Sciences", 2010. pp.1644-1649

[4] Min Wu, Heather Yu, Bede Liu (2003, June). Data Hiding in Image and Video: Part II – Designs and Applications. IEEE Trans. Image Processing. 12(6)

[5] Esen E. Alatan A.A. (2011, August) Robust Video Data Hiding Using Forbidden Zone Data Hiding and Selective Embedding. IEEE Transactions on Circuits and Systems for Video Technology, 21(8)

[6] Biswajit Biswas."Digital Watermarking for MPEG video", Tata elxsi engineering creativity.

[7] Spyridon K. Kapotas, Eleni E. Varsaki and Athanassios N. Skodras, "Data Hiding in H.264 Encoded Video Sequences", in Proc.IEEE 9th Workshop Multimedia Signal Process, Oct 2007, pp. 373-376

[8] Anindya Sarkar, Upamanyu Madhow, Shivkumar Chandrasekaran and Bangalore. S. Manjunath, "Adaptive MPEG-2 Video Data Hiding Scheme", in Proc. 9th SPIE Security Steganography Watermarking Multimedia Contents, 2007, pp. 373-376

[9] K. Solanki, Noah Jacobsen, U. Madhow, B. S. Manjunath and Shivkumar Chandrasekaran, (2004, December) "Robust Image-Adaptive Data Hiding Using Erasure and Error Correction", IEEE Trans. Image. Process, 13(12), pp. 1627-1639

[10] Venkata Pavan Kumar, Dr. C.P.V.N.J. Mohan Rao, Satya P Kumar Somayajula (2012, September), Data Hiding Using Bit-Stream Level Through Selective Embedding and Modulation. International Journal of Engineering Science and Technology. 2(5). Pp. 1377-1385

[11] Xinpeng Zhang (2011, April). Data Hiding in Encrypted Image. IEEE Signal Processing Letters, 18(4)

[12] Kaushal M. Solanki, 2005, Multimedia Data Hiding: From Fundamental Issues to Practical Techniques