Embedding Encrypted Data in Video using Symmetric Key Cryptography

Shraddha Korde, Bhavik Jethwa, Ranjit Kundaram, B.W.Balkhande

Abstract*— *Video data hiding is still an important research topic due to the design complexities involved. We propose a new video encrypted data hiding method that makes use of erasure correction capability of repeat accumulate codes and superiority of forbidden zone data hiding as well as DNA cryptography logic is used for encryption and decryption of the data. This paper also proposes a unique cipher text generation procedure as well as a new key generation procedure. DNA cryptography is one of the major concerned areas of computer and data security and a very promising direction in cryptography research. Selective embedding is utilized in the proposed method to determine host signal samples suitable for data hiding. This method also contains a temporal synchronization scheme in order to withstand frame drop and insert attacks. Finally, to demonstrate the performance of the proposed method, its implementation is explained and the results are analyzed.

Index Terms*— *Cipher text, Data hiding, decryption, encryption, forbidden zone data hiding, key generation, repeat accumulate codes, selective embedding, security.

I. INTRODUCTION

Recent research trends have focused on introducing DNA medium so as to obtain complex computation in the process of achieving the cipher text. DNA cryptography is the new field of interest in the common PKI scenario, where it is possible to follow the pattern of PKI, while also exploiting the inherent massively parallel computing properties of DNA bonding to perform the encryption and decryption of the public and private keys**.** Public Key Cryptography is one set of cryptographic techniques for providing confidentiality, preventing data compromise, detecting alteration of data and verifying its authenticity.

Essential parts of what we may call data security, specifically confidentiality and authentication, are achieved using cryptography [3].

In this paper, the proposed algorithm takes its basic idea from the way DNA encodes the genetic information in the codons (i.e. each codon holds the information of a particular protein to be synthesized). So using this idea any plaintext can be encoded with a one-time code book. In figure1, for the plain text “ROHAN” the cipher text could be “SDRDWEFGFD”. With this idea, a substitution algorithm is being proposed that is discussed in this paper. The scheme is principally a symmetric key algorithm, except that the sender initially has only part of the keys, and he generates the rest part of the keys. Symmetric key based encryption is the only way for secure communication between nodes.

Manuscript received on April 16, 2014.

Shraddha Korde, Computer Engineering, Bharati Vidyapeeth College of Engineering, Navi Mumbai, India.

Bhavik Jethwa, Computer Engineering, Bharati Vidyapeeth College of Engineering, Navi Mumbai, India.

Ranjit Kundaram, Computer Engineering, Bharati Vidyapeeth College of Engineering, Navi Mumbai, India.

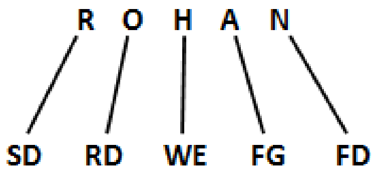

Figure 1: Example of one time code block

Data hiding is the process of embedding information into a host medium. In general, visual and aural media are preferred due to their wide presence. Although the general structure of data hiding process does not depend on the host media type, the methods vary depending on the nature of such media. For instance, image and video data hiding share many common points.

Data hiding in video sequences is performed in two major ways: bit stream-level and data-level. In bit stream-level, the redundancies within the current compression standards are exploited. Typically, encoders have various options during encoding and this freedom of selection is suitable for manipulation with the aim of data hiding. However, these methods highly rely on the structure of the bit stream; hence, they are quite fragile, in the sense that in many cases they cannot survive any format conversion or transcoding, even without any significant loss of perceptual quality. As a result, this type of data hiding methods is generally proposed for fragile applications, such as authentication. On the other hand, data level methods are more robust to attacks. Therefore, they are suitable for a broader range of applications.

However, most of the video data hiding methods utilize uncompressed video data. Sarkar *et al.* [7] proposed a high volume transform domain data hiding in MPEG-2 videos. They applied quantization index modulation (QIM) to low frequency DCT coefficients and adapted the quantization parameter based on MPEG-2 parameters. Furthermore, they varied the embedding rate depending on the type of the frame. As a result, insertions and erasures occur at the decoder, which causes de-synchronization. They utilized repeat accumulate (RA) codes in order to withstand erasures.

We propose a new block-based selective embedding type data hiding framework that encapsulates forbidden zone data hiding (FZDH) [9] and RA codes in accordance with an additional temporal synchronization mechanism. RA codes allow handling de-synchronization between embedder and decoder that occurs as a result of the differences in the selected coefficients. In order to incorporate frame synchronization markers, we partition the blocks into two groups. One group is used for frame marker embedding and the other is used for message bits. We utilize systematic RA codes to encode message bits and frame marker bits. Each bit is associated with a block residing in a group of frames. Random interleaving is performed spatio-temporally; hence, dependency on local characteristics is reduced. The block is selected using a technique which is explained later. The unselected blocks are labeled as erasures and they are not processed. For each selected block, there exists variable number of coefficients. These coefficients are used to embed and decode single message bit by employing multidimensional form of FZDH that uses cubic lattice as its base quantizer.

II. PROPOSED SYSTEM

The proposed symmetric key cryptography method has introduced a new format of cipher text, where the primary cipher text obtained after encoding is being divided into three unequal parts and then extra parameters such as primer code, file type code, integrity code, and authentication code are added in between parts of the cipher text to obtain the final cipher text.

Abbreviations used are: CT-PRIMARY CIPHER TEXT, AUT-AUTHENTICATIONCODE, INTR-INTEGRITY CODE, FT-FILE TYPE CODE, SPM-STARTING PRIMER (GARBAGE), EPM-ENDING PRIMER (GARBAGE) &OCT-ORIGINAL CIPHER TEXT FORMAT.

A. Format of Cipher Text

From plain text (PT) the primary cipher text (CT) is obtained by using the encryption algorithm and the 1st level key (PK1). The following steps are to be followed to obtain the final cipher text.

Step-1: Encrypt the plain text with 1st level key (PK1).

Step-2: Divide the primary cipher into three unequal parts (Fig-2)

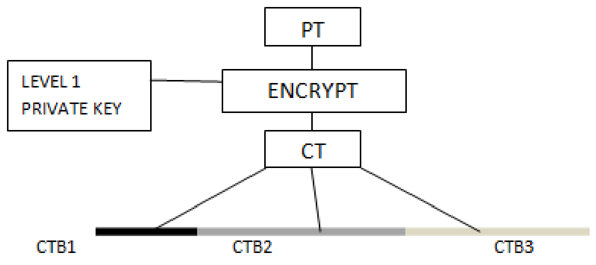

Figure 2: Divide the cipher into three unequal parts

Step-3: Attach AUT, INTR, FT, SPM, EPM with the above CTB's as follows (fig-3) after encrypting CTB's using level2

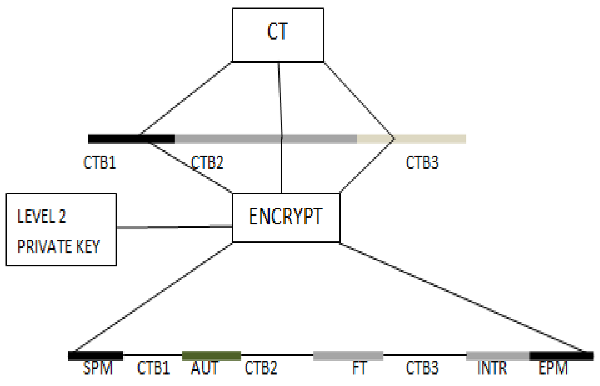

Figure 3: Original cipher format

private keys (which include the information about the introns(AUT, FT, ETC) positions and the length of the SPM and EPM).

B. Procedure Level1PrivateKeyGeneration

Sender's side computation:

Step 1: First the receiver will send a number as public key (PK) through private channel or public channel. This key should be between the ranges 1 to 255.

Step 2: Sender will generate one random number (R).

Step 3: The random number selected is being represented in binary and then its complement is being again converted into decimal which will be used as the Encryption key (E).

(For e.g.: let the Public Key is PK=7, and the random number R=5.Binary representation of R = 101 (4-bit). Complement of R = 010. Therefore, In Decimal R= 2. This 2 will be used as Encryption Key (i.e. .E=2)).

Step4: sender will compute the level1 private key as follows:

Remainder computation (r):

(PK * R) % 16 = (7*5) % 16 =3 (Remainder)

Hexadecimal Notation = 3

Quotient computation (c):

(PK * R) / 16 = 35 / 16 =2, Hexadecimal Notation = 2

Concatenating these two hexadecimal notations,

*We get **rc** = 32.*

Step-5: Sender will send ***rc*** as level1 private key through private Channel with level2 private keys. These two keys (level1 & level2) are sending in a digest form (in progress) through private channel.

Receiver's side computation:

Step 1: Receiver will receive 32 & separate the numbers r and c and convert these to equivalent decimal notation.

Step2: Receiver will compute the decryption key as follows:

Decimal value computation (X):

X= (16 * c) + r (e.g. X= (16 * 2) + 3 = 35)

Intermediate key computation (K1):

K1= (X / PK) (e.g. K1= (35 / 7) = 5, where 7 is PK)

Step-3: Convert 5 to binary form and complement it.

(E.g. binary of 5=101= 010 =2 in Decimal Notation)

Step 4: Therefore, 2 is the level1 private key (PK1) to be used for decryption.

C. Procedure Encryption

Step 1: Let, Q be an array of size 16 and R be an array of size 16 also. For example,

Q= {'A','B','C'…'Q'} R= {'/','#','$'…}

Step 2: Input the file name with its extension. e.g. abc024.txt. (*Plain Text*)

Step 3: Convert the file into its' corresponding byte codes (the range of the byte codes will be from -128 to+127).

Step 4: In order to get the index of the arrays we have to change the negative value byte codes into positive values by adding +128 to each of the byte values.

For e.g.: -120 become +8.Thus the range of the byte codes becomes 0 to 255.

Step 5: Each of the byte will be taken in account of calculation as n1= (byte_code / 16) and n2= (byte_code%16).

For e.g.: if 92 is the byte code then n1= 92 / 16 =5 and n2= 92%16= 12

Step 6: Now the key will be added up with the numbers n1 &

n2 to get the new indexes q and r as
q= [(n1+k1) %16] and r= [(n2+ k1) % 16], where k1 is the key value.
For example, Let the key is k1=7.So,
q = (n1+7) %16 = (5+7) %16=12%16=12
r = (n2+7) %16= (12+7) %16=19%16=3
Step 7: The numbers q and r will be used as the index of the static arrays Q & R. For example,
Q= {'A','B'….'L'…'Q'} R={'/','#','%','?'…}
q= {0, 1 …… 12…15} r= {0, 1, 2, 3...}

$$\begin{tikzcd} {} \arrow[d] & & {} \arrow[d] \\ L \arrow[dr] & & ? \arrow[dl] \\ & L? & \end{tikzcd}$$

Thus the byte code '92' is converted into 'L?'
Step 8: After getting the cipher of each byte code, concatenate all byte codes in order to get the cipher text.
Step 9: This cipher text will be written into a text file & send this file to the receiver end.

D. Procedure Decryption

Step 1: Read the input cipher file, two bytes at a time.
Step 2: Split the code. For example,

$$\begin{tikzcd} & L? \arrow[dl] \arrow[dr] & \\ L & & ? \end{tikzcd}$$

Step 3: Search the array Q to get the index of 'L' and array R to get the index of '?'. Therefore we will get the index of 'L'=12 and the index of '?'=3.
Step 4: Subtract the key from each of the index value. If the result becomes negative, add 16 with it.
For example, n1=12–7=5 and n2= 3–7=-4. Since n2<0, so by adding 16 with it, we can get n2= -4+16 = 12.
Step 5: Now multiply n1 by 16 and add n2 with it to get the byte code. So Byte code = (n1*16) + n2.
e.g. (5*16) +12 = 92.
Step 6: Convert the byte code '92' into its corresponding ASCII code.
Step 7: Save the file with the extension.

III. LEVEL TWO KEY GENERATION PROCEDURE

The level2 private key gives the information about the length of the primer & the positions of introns (fixed length garbage text, AUT, SPM, FT, EPM, INTR). The primers are added at the starting and at the ending of primary cipher text (CT); introns are inserted within the cipher text at positions as described by the Level (2) key. The sender's file length is chosen as level2 key. The sum of the digits of the sender's file length is taken as the input to decide the primer's length. Introns positions are taken on the basis of the individual digits of the file length. The sender sends file length using the following procedure:
Step 1: First the receiver will send a number through private channel or public channel. This key should be between the ranges 1 to 255. Now the sender will perform the following task using this number.
Step 2: Let P be an array which will hold the secondary level of keys.
Step 3: Take a variable and initialize it with a number which is the file length.
Step 4: Repeat through the following steps for 1 to number of digits in N.
Step 5: Perform digit wise X – OR of N from left to right (i.e. from MSB to LSB) (Fig-4). It is done in the following manner:
Step 6: N = rn-1 rn – 2…….r1.
Step 7: PK = rn
Step 8: P[i++] = rn.
Step 9: Send the array P (level2 private key) to the receiver to get the file length by applying reverse procedure of the above.

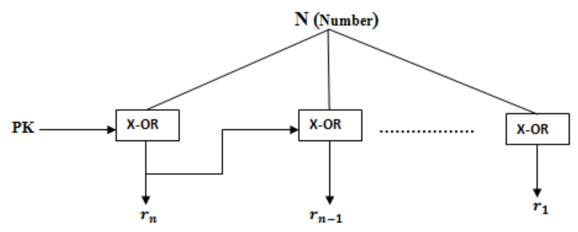

Figure 4 : XOR operation of the PK

IV. PROPOSED VIDEO DATA HIDING

We propose a block based adaptive video data hiding method that incorporates FZDH, which is shown to be superior to QIM and competitive with DC-QIM [9], and erasure handling through RA Codes. We utilize selective embedding to determine which host signal coefficients will be used in data hiding as in [8]. The de-synchronization due to block selection is handled via RA Codes as in [7] and [8]. Furthermore, as in [5], we equip the method with frame synchronization markers in order to handle frame drop, insert, or repeat attacks. Hence, it can be stated the original contribution of this paper is to devise a complete video data hiding method that is resistant to de-synchronization due to selective embedding and robust to temporal attacks, while making use of the superiority of FZDH.

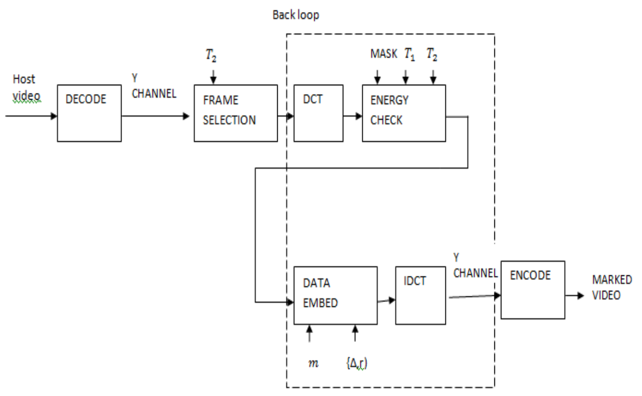

Figure 5: Embedder flowchart of the proposed video data hiding framework for a single frame.

A. Framework

The embedding operation for a single frame is shown in Fig. 6. Y-channel is utilized for data embedding. In the first step, frame selection is performed and the selected frames are processed block-wise. For each block, only a single bit is hidden. After obtaining 8 × 8 DCT of the block, energy check is performed on the coefficients that are predefined in a mask. Selected coefficients of variable length are used to hide data bit m. m is a member of message bits or frame synchronization markers. Message sequence of each group is obtained by using RA codes for T consecutive frames. Each block is assigned to one of these groups at the beginning. After the inverse transform host frame is obtained. Decoder is the dual of the embedder, with the exception that frame selection is not performed. Fig. 6 shows the flowchart for a single frame. Marked frames are detected by using frame synchronization markers. Decoder employs the same system parameters and determines the marked signal values that will be fed to data extraction step. Erasures and decoded message data probabilities (om) are passed to RA decoder for T consecutive frames as a whole and then the hidden data is decoded.

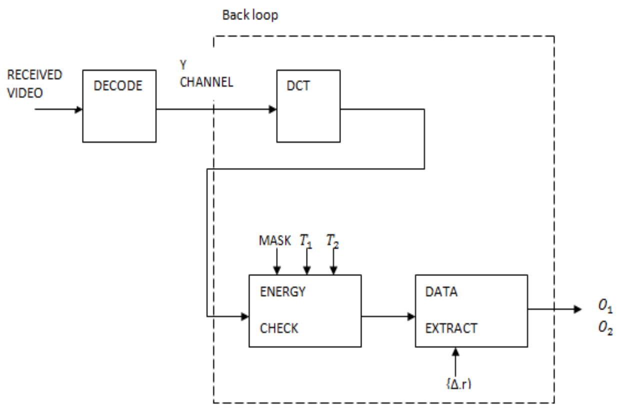

Figure 6: Decoder flowchart of the proposed video data hiding framework for a single frame.

B. Selective Embedding

Host signal samples, which will be used in data hiding, are determined adaptively. The selection is performed at four stages:

1) Frame selection: selected number of blocks in the whole frame is counted. If the ratio of selected blocks to all blocks is above a certain value (T0) the frame is processed. Otherwise, this frame is skipped.

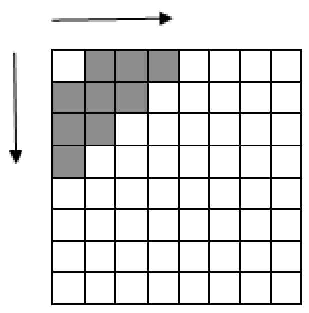

Figure 7: Sample coefficient mask denoting the selected frequency band.

2) Frequency band: only certain DCT coefficients are utilized. Middle frequency band of DCT coefficients shown in Fig. 7 is utilized similar to [7].

3) Block selection: energy of the coefficients in the mask is computed. If the energy of the block is above a certain value (T1) then the block is processed. Otherwise, it is skipped.

4) Coefficient selection: energy of each coefficient is compared to another threshold T2. If the energy is above T2, then it is used during data embedding together with other selected coefficients in the same block.

C. Block Partitioning

Two disjoint data sets are embedded: message bits (m1) and frame synchronization markers (m2). The block locations of m2 are determined randomly depending on a random key. The rest of the blocks are reserved for m1. The same partitioning is used for all frames. m2 is embedded frame by frame. On the other hand, m1 is dispersed to T consecutive frames. Both of them are obtained as the outcomes of the RA encoder.

D. Erasure Handling

Due to adaptive block selection, de-synchronization occurs between embedder and decoder. As a result of attacks or even embedding operation decoder may not perfectly determine the selected blocks at the embedder. In order to overcome this problem, error correction codes resilient to erasures, such as RA codes are used in image [8] and video [7] data hiding in previous efforts. RA code is a low complexity turbo-like code. It is composed of repetition code, interleaver, and a convolutional encoder. The source bits (u) are repeated R times and randomly permuted depending on a key. The interleaved sequence is passed through a convolutional encoder with a transfer function 1/(1 + D), where D represents a first-order delay.

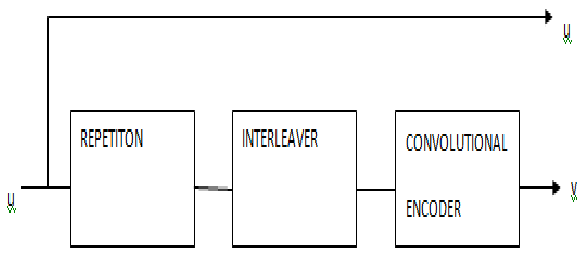

Figure 8: RA encoder (u denotes source bits and u + v denote encoded bits).

In systematic RA code, input is placed at the beginning of the output as shown in Fig. 8. In this paper, we utilize systematic RA codes to obtain m1 as u1+v1 and m2 as u2+v2. Here, u1 denotes the uncoded message bits and u2 is the uncoded frame synchronization marker bits. RA code is decoded using sum-product algorithm. We utilize the message passing algorithm given in [6].

E. Frame Synchronization Markers

Each frame within a group of T consecutive frames is assigned a local frame index starting from 0 to T −1. These markers are used to determine the frame drops, inserts and repeats, as well as the end of the group of frames at which point all necessary message bits are available for RA decoder. RA code spreads the output codeword’s of the adjacent frame indices; hence, errors are less likely to occur when decoding adjacent frame indices.

F. Soft Decoding

At each frame, frame synchronization markers are decoded first. Message decoding is performed once the end of the group of frames is detected. Two frame index values are stored: current and previous indices. Let fcur and fpre denote the current and previous frame indices, respectively. Then the following rules are used to decode u1.

1) If fcur >T, then skip this frame. (This case corresponds to unmarked frame).
2) If fcur = fpre, then skip this frame. (This case corresponds to frame repeat).
3) Otherwise, process the current frame.. Non- selected blocks are left as erasures.

If fcur<fpre, then the end of the group of frames is reached. Decode the message bits and obtain u1. Initialize data structure.

V. PERFORMANCE ANALYSIS OF THE PROPOSAL

The proposed procedures are implemented in java platform for its platform independent property and available in-built cryptography functionalities. The procedures are implemented successfully for the specified sized input plain texts. Initially, there were constraints for large files such as images or videos where the required primary memory of the system could create a problem in the execution and conversion of the plain text into cipher text. But, later on that problem are also resolved by simply dividing the large file into fixed sized sub-files and then performing the swapping while encoding and decoding. The following table represents the data sets that are obtained during the testing and analysis of the proposed procedure.

Table -1: Datasets for performance analysis

TESTS	File size(KB)	Cipher size(KB)	Encrypt time(ms)	Decrypt time(ms)
TEST1	1126.4	2252.8	9531	8672
TEST2	2304	4608	18966	11375
TEST3	5990.4	11980.8	46936	28406
TEST4	14950	29900.8	119281	61500
TEST5	35840	71680	312860	136625
TEST6	57856	115712	477687	256453

VI. CONCLUSION

In this paper, we proposed a new video data hiding framework that makes use of erasure correction capability of RA codes and superiority of FZDH. The method is also robust to frame manipulation attacks via frame synchronization markers. The proposed cryptography procedure has included only the concept DNA cryptography and as DNA computing has become a large area of interests in the research domain of cryptography, it increases the security technologies of encryption, steganography, signature and authentication by using DNA molecular as information medium. There are a lot of opportunities in expanding and manipulating DNA characteristics and operations to solve real application especially industrial engineering and management engineering problems. Although this method is efficient, and it is powerful against certain attacks; the partial information contained in the cipher text makes the method much stronger.

REFERENCES

1. B. Roy, G. Rakshit, P. Singha, A. Majumder and D. Datta, “An improved Symmetric key cryptography with DNA based strong cipher”, Department of Computer Science and Engineering Tripura Institute of Technology, Narsingarh, Tripura, India.
2. Ersin Esen and A. Aydin Alatan, “Robust Video Data Hiding Using Forbidden Zone Data Hiding and Selective Embedding” in *IEEE transactions on circuits and systems for video technology*, vol. 21, no. 8, august 2011.
3. Garfinkel Simson, Web Security, Privacy & Commerce, 2nd Edition, O’Reilly Publisher, November 2001.
4. M. Wu, H. Yu, and B. Liu, “Data hiding in image and video: I. Fundamental issues and solutions,” *IEEE Trans. Image Process.*, vol. 12, no. 6, pp. 685–695, Jun. 2003.
5. M. Wu, H. Yu, and B. Liu, “Data hiding in image and video: II. Designs and applications,” *IEEE Trans. Image Process.*, vol. 12, no. 6, pp. 696–705, Jun. 2003.
6. M.M.Mansour,“A turbo-decoding message-passing algorithm for sparse parity-check matrix codes,”*IEEE Trans. Signal Process.*,vol. 54, no. 11,pp. 4376–4392, Nov. 2006.
7. A. Sarkar, U. Madhow, S. Chandrasekaran, and B. S. Manjunath, “Adaptive MPEG-2 video data hiding scheme,” in *Proc. 9th SPIE Security Steganography Watermarking Multimedia Contents*, 2007, pp. 373–376.
8. K. Solanki, N. Jacobsen, U. Madhow, B. S. Manjunath, and S. Chandrasekaran, “Robust image-adaptive data hiding using erasure and error correction,” *IEEE Trans. Image Process.*, vol. 13, no. 12, pp. 1627–1639, Dec. 2004.
9. E. Esen and A. A. Alatan, “Forbidden zone data hiding,” in *Proc. IEEE Int. Conf. Image Process.*, Oct. 2006, pp. 1393–1396.