Data Hiding in Videos Using Codeword Substitution

Nerin Thomas

***Abstract*- Data hiding is one of the important techniques in the emerging world for reducing the increased attacks. It is also known as data encapsulation or information hiding and is mainly used for hiding internal object details. In order to maintain security and privacy, digital video sometimes needs to be stored and processed in an encrypted format. The performance of data hiding in these encrypted videos is very necessary for the purpose of content notation or tampering detection. Without decryption, data hiding in these encrypted videos will protect the confidentiality of the content. The capacity for performing the data hiding directly in these encrypted H.264/AVC video stream can eliminate the leakage of video content and can help the privacy concerns with cloud computing. In this paper, it proposes a new method to embed secret data directly in the encrypted H.264/AVC bit stream. It can have the following three parts: H.264/AVC video encryption, data embedding, and data extraction. Here, the data extraction can be done either in the encrypted or in the decrypted domain in order to adapt to different application scenarios.**

***Index Terms*- Data hiding, encryption, decryption, H.264/AVC video, codeword substitution**

I. INTRODUCTION

Now a days, cloud computing has become an important emerging technology and can provide highly efficient computation and large-scale storage solution for videos. However, the cloud services may attract more attacks and are vulnerable to untrustworthy system administrators. Here, it proposes a new method to embed secret data directly in compressed and encrypted H.264/AVC bit stream. By analyzing the property of H.264/AVC codec, the codeword's of IPMs, the codeword's of MVDs, and the codeword's of residual coefficients, firstly the data's are encrypted with a stream cipher. To keep the codeword length unchanged, the encryption algorithm is combined with the Exp-Golomb entropy coding and Context-adaptive variable-length coding (CAVLC). Because of the privacy-preserving requirements from cloud data management, data hiding in encrypted media is a new topic for detecting attacks. It is simple to implement as it is directly performed in the compressed and encrypted domain, this algorithm can preserve the bit-rate exactly even after encryption and data embedding.

Manuscript received January 30, 2015.
Nerin Thomas, Computer Science Department, M. G. University, Pathanamthitta, India, 9495772071.

The diagram for the proposed framework is depicted below in which the encryption and data embedding are depicted in part one, and the data extraction and video decryption are depicted in part two.

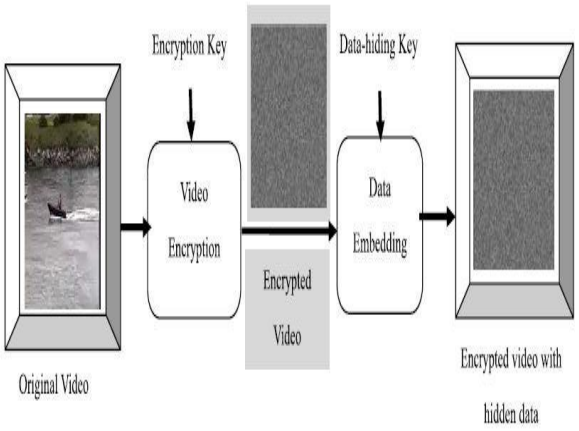

Fig: video embedding and data extraction at the sender end

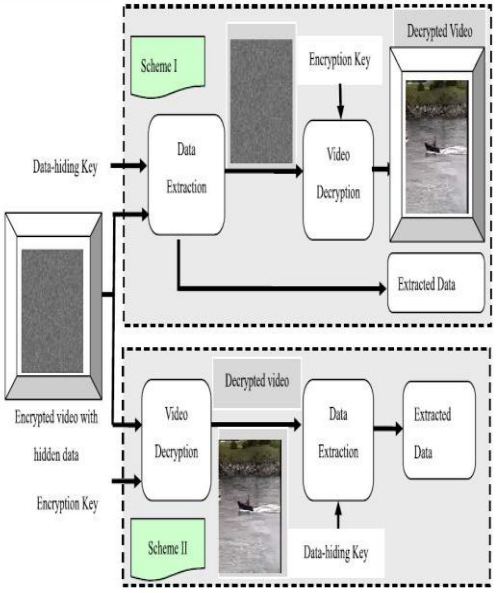

Fig: Data extraction and video display at the receiver end in two scenarios

II.WORKING

Here, the main aim of the content owner is to use the standard stream ciphers with encryption keys to encrypt the original H.264/AVC video stream in order to produce an encrypted video stream. By using codeword substitution, the data hider can then add the additional data into the encrypted video stream without the original video content knowledge. The hidden data extraction can be accomplished either in encrypted or in decrypted version at the receiver end.

A.Intra-Prediction Mode (IPM) Encryption

Four types of intra coding are supported in H.264/AVC standard which can be denoted as Intra 4 × 4, Intra 16×16, Intra chroma, and I PCM. In this method, IPMs in the Intra 4×4 and Intra 16 × 16 blocks are selected for encryption.

In the Intra 16 × 16, four intra prediction modes (IPMs) are available. The IPM for Intra 16 × 16 blocks is specified in the mb_type (macro block type) field which also specifies other parameters about this block such as coded block pattern (CBP).

mb_type	Name of mb_type	Intra16x16 PredMode	Chroma CBP	Luma CBP	Codeword
1	I_16x16_0_0_0	0	0	0	010
2	I_16x16_1_0_0	1	0	0	011
3	I_16x16_2_0_0	2	0	0	00100
4	I_16x16_3_0_0	3	0	0	00101
5	I_16x16_0_1_0	0	1	0	00110
6	I_16x16_1_1_0	1	1	0	00111
7	I_16x16_2_1_0	2	1	0	0001000
8	I_16x16_3_1_0	3	1	0	0001001
9	I_16x16_0_2_0	0	2	0	0001010
10	I_16x16_1_2_0	1	2	0	0001011
11	I_16x16_2_2_0	2	2	0	0001100
12	I_16x16_3_2_0	3	2	0	0001101
13	I_16x16_0_0_1	0	0	15	0001110
14	I_16x16_1_0_1	1	0	15	0001111
15	I_16x16_2_0_1	2	0	15	000010000
16	I_16x16_3_0_1	3	0	15	000010001
17	I_16x16_0_1_1	0	1	15	000010010
18	I_16x16_1_1_1	1	1	15	000010011
19	I_16x16_2_1_1	2	1	15	000010100
20	I_16x16_3_1_1	3	1	15	000010101

Table: Macro block Types for I Slices and Variable Length of Codeword in H.264/AVC

Here, it can encrypt the codeword of an IPM without modifying the CBP in order to maintain the standard-compliance of the encrypted bit stream. Along with this, the encrypted codeword should have the same size as the original codeword for keeping the codeword's length unchanged. From the table shown above, it can be observed that the combination of CBP is the same in every four lines, and the codeword's have the same length in every two consecutive lines. For example, the corresponding codeword's of "5" and "6" are "00110" and "00111", respectively and can have the same length and the same value of CBP. Thus the IPM encryption is performed in Intra 16 × 16 blocks by applying a bitwise XOR operation between the last bits of the codeword's and a bit of the pseudorandom sequence in order to keep the value of CBP and the length of codeword unchanged. Through a standard secure cipher determined by an encryption key E_Key1; the pseudorandom sequence can be generated.

In H.264/AVC, it can have nine prediction modes (0-8) for Intra 4 × 4 luminance blocks. For each Intra 4× 4 luminance blocks, the prediction mode should be signaled to the decoder and this could require a large number of bits. The predictive coding technique can be applied to signal prediction modes for compressing the prediction-mode data. The most probable mode (MPM_E) is defined as the smaller of the prediction modes of the spatially adjacent upper block A and left block B. $Mode_E$ is the prediction mode of the currently considered block E. Only one bit is needed to signal the prediction mode when the $Mode_E$ is equal to MPM_E. The codeword is composed of one flag bit "0"and three bits fixed-length code when $Mode_E$ and MPM_E are different.

If ModeE is equal to MPME then the codeword is kept unchanged for Intra_4 × 4 blocks. If do not, through a standard secure cipher determined by an encryption key E_Key2, three bits fixed-length code (denoted as X)in each codeword is encrypted with a pseudorandom sequence which is generated. Bitwise XOR operation can be used for the encryption scheme.

B. Motion Vector Difference (MVD) Encryption

The motion vectors should be encrypted In order to protect both texture information and motion information. It yields MVD through motion vector prediction which is further performed on the motion vectors. To encode MVD, Exp-Golomb entropy coding can be used.

The values of MVDs and corresponding Exp-Golomb codeword's can be shown in the table given below. By applying the bitwise XOR operation with a standard stream cipher, which is determined by an encryption key E_Key3, the last bit of the codeword is encrypted.

MVD	*code_num*	*codeword*
0	0	1
1	1	010
-1	2	011
2	3	00100
-2	4	00101
3	5	00110
-3	6	00111
4	7	0001000
-4	8	0001001
5	9	0001010
-5	10	0001011
6	11	0001100
-6	12	0001101
7	13	0001110
-7	14	0001111
8	15	000010000
-8	16	000010001
9	17	000010010
-9	18	000010011
…	…	…

Fig: MVDs and Corresponding EXP-GOLOMB Codeword's

For example, the corresponding codeword's of "3" and "–3" are "00110" and "00111" can have the same length. During the encryption process, when the value of MVD is equal to 0, its corresponding codeword "1" remains unchanged.

C. Residual Data Encryption

Another type of sensitive data called the residual data in both I-frames and P-frames should be encrypted for keeping high security. In order to encode the quantized coefficients of a residual block, it uses the CAVLC entropy coding. By the following format, we can express each CAVLC codeword like:

{Coeff token, Sign of Trailing Ones, Level, Total zeros, Run before}

During encryption process, not all syntax elements can be modified in order to keep the bit stream compliant. Coeff token, Total zeros, and Run before should remain unchanged. By modifying the codeword's of Sign_of_TrailingOnes and Level, residual data encryption can be accomplished.

With single bit, the Sign_of_TrailingOnes can be encoded. For +1, bit "0" is assigned and for −1, bit "1" is assigned. With a standard stream cipher which is determined by an encryption key E_Key4 and by applying the bitwise XOR operation, the codeword of Sign_of_TrailingOnes can be encrypted. Levels with different suffix Length and corresponding codeword's will be shown in the below table.

suffixLength	*Level*(>0)	*Codeword*	*Level*(<0)	*Codeword*
0	1	1	-1	01
	2	001	-2	0001
	3	00001	-3	000001
	4	0000001	-4	00000001
1	1	10	-1	11
	2	010	-2	011
	3	0010	-3	0011
	4	00010	-4	00011
	5	000010	-5	000011
	6	0000010	-6	0000011
	7	00000010	-7	00000011
	8	000000010	-8	000000011
2	1	100	-1	101
	2	110	-2	111
	3	0100	-3	0101
	4	0110	-4	0111
	5	00100	-5	00101
	6	00110	-6	00111
	7	000100	-7	000101
	8	000110	-8	000111
	9	0000100	-9	0000101
	10	0000110	-10	0000111
	11	00000100	-11	00000101

Table:Levels and Corresponding Codeword's

For example, if the suffix Length is equal to 2, the codeword's corresponding to "3" and "-3" is "0100" and "0101" and can have the same length. Along with this, during the encryption process, when suffix Length is equal to 0 then the codeword's should remain unchanged.

For each Level, the codeword should be made up of a prefix (level prefix) and a suffix (level suffix) as:

Level codeword = [level prefix], [level suffix]

D. Data Embedding

The following three limitations should satisfy the codeword's substitution. After codeword substitution, the bit stream must remain syntax compliance in order to decode with a standard decoder is the first one. Second is the substituted codeword should have the same size as the original codeword in order to keep the bit-rate unchanged. Third is that the impact should be kept to minimum bt the data hiding does cause visual degradation. For data hiding the codeword's of Levels within P-frames are used whereas the codeword's of Levels in I-frames are remained.

No corresponding substituted codeword's are present, when suffix Length is equal to 0 or 1. We cannot find a pair of codeword's with the same size when suffix Length is equal to 0. One codeword also cannot be substituted by another codeword with the same size, since this substitution would change the sign of Level when suffix Length is equal to 1. The suffix Length is 2 or 3 of the codeword's of Levels would be divided into two opposite code spaces can be denoted as C0and C1 which is shown below. With binary hidden information "0" and "1", the codeword's assigned in C0and C1 are associated.

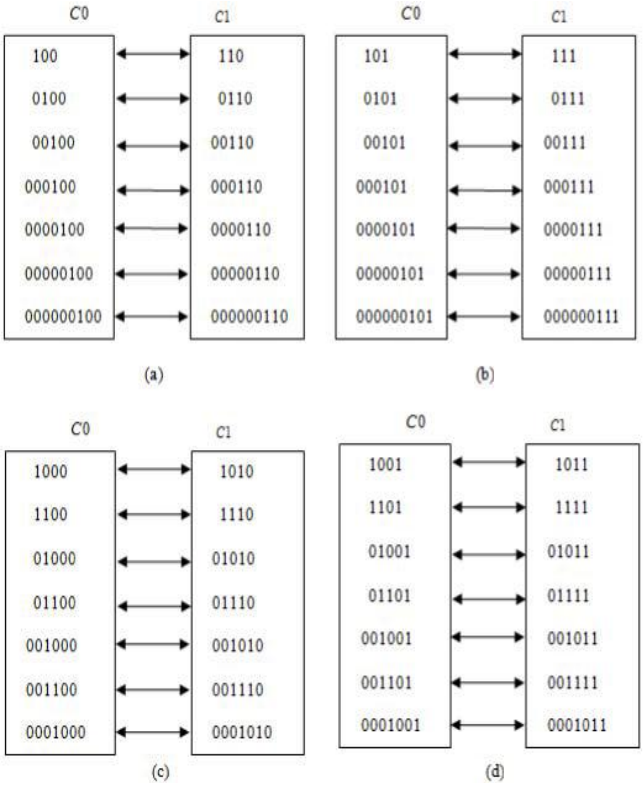

Fig: CAVLC codeword mapping. (a) Suffix Length = 2& Level > 0. (b) Suffix Length = 2& Level > 0. (c) Suffix Length = 3& Level > 0. (d) Suffix Length = 3& Level < 0.

We want to embed the additional data is a binary sequence denoted as B = {b (i) |i = 1, 2, • • •, L, b(i) ∈ {0,1}}. Through the following steps, data hiding can be performed in the encrypted bit-Stream directly:

Step1. The additional data is encrypted with the chaotic pseudo-random sequence in order to enhance the security = {p(i)|i = 1,2,••• , L, p(i) ∈ {0,1}}[22] to generate the to-be-embedded sequence W = {w(i)|i = 1, 2, • • • , L , w(i) ∈ {0,1}}. By using logistic map with an initial value, i.e., the data hiding key, the sequence P is generated. To recover the additional data, it is very difficult for anyone who does not retain the data hiding key.

Step2. By parsing the encrypted H.264/AVC bit stream, the codeword's of Levels can be obtained.

Step3. By codeword substitution, the data bit can be embedded only if the current codeword belongs to code spaces C0orC1. The codeword is left unchanged otherwise.

The codeword substitutions procedure for data hiding is shown in the figure.

```
Procedure
  if (data bit= =0)
  {
      if (the codeword belongs to C0)
          The codeword is unmodified;
      else if (the codeword belongs to C1)
          The codeword is replaced with the corresponding codeword in C0.
  }
  else if (data bit= =1)
  {
      if (the codeword belongs to C1)
          The codeword is unmodified;
      else if (the codeword belongs to C0)
          The codeword is replaced with the corresponding codeword in C1.
  }
```

Fig: The procedure of codeword mapping.

Step4. For data hiding, select the next codeword and then go to Step3. The embedding process is stopped when there are no more data bits to be embedded.

E. Data Extraction

In data extraction, either in encrypted or decrypted domain the hidden data can be extracted. It is fast and simple.

1) Scheme I: Encrypted Domain Extraction. A database manager may only get access to the data hiding key and have to manipulate data in encrypted in order to protect privacy. Encrypted domains data extraction ensures the feasibility of our scheme in this case. Here, to the data extraction module with hidden data the encrypted video is directly sent and the extraction process can be done as follows:

Step1: By parsing the encrypted bit stream, the codeword's of Levels are firstly identified.

Step2: The extracted data bit is "0", when the codeword belongs to code space C0. The extracted data bit is "1", if the codeword belongs to code space C1.

Step3: The same chaotic pseudo-random sequence P that was used in the embedding process can be generated on the basis of the data hiding key. By using P, the extracted bit sequence could be decrypted in order to get the original additional information. It significantly avoids the leakage of original video content because the whole process is entirely operated in encrypted domain.

2) Scheme II: Decrypted Domain Extraction. Both embedding and extraction of the data are performed in encrypted domain in scheme I. Users wants to decrypt the video first and extract the hidden data from the decrypted video. For example, an authorized user with hidden data who owned the encryption key received the encrypted video. Using the encryption key, then the received video can be decrypted. The decrypted video with the hidden data can be used to trace the source of the data.

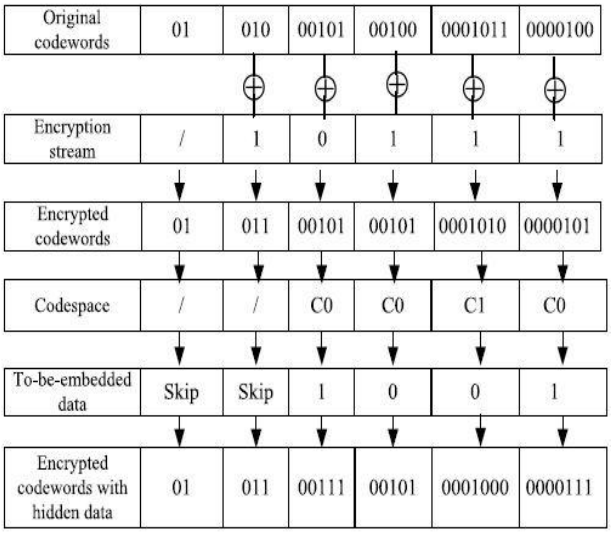

Fig: Data Embedding

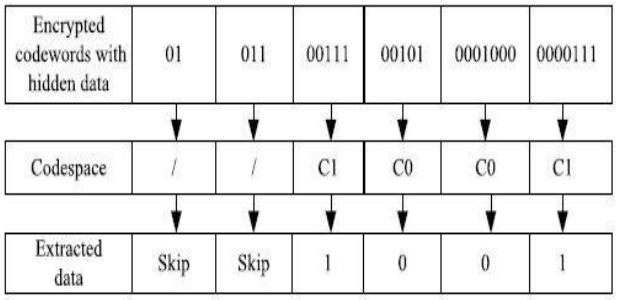

Fig: Data Extraction in encrypted domain

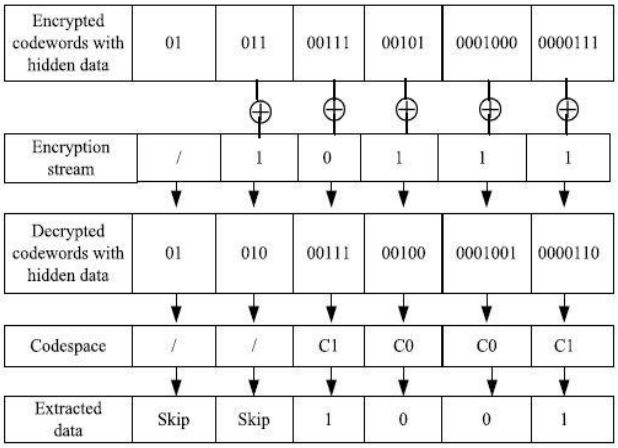

Fig: Data Extraction in decrypted domain

The decryption and data extraction's whole process can be given as follows:

Step1: With the encryption keys as given in encryption process, generate encryption streams.

Step2: By parsing the encrypted bit stream, identify the codeword's of IPMs, MVDs, Sign_of_TrailingOnes and Levels.

Step3: Because the XOR operation is symmetric, the decryption process is identical to the encryption process. By performing XOR operation with generated encryption streams, the encrypted codeword's can be decrypted and then two XOR operations cancel each other out which renders the original plaintext. The decryption is possible only for the authorized users since the encryption streams depend on the encryption key. The owner can further extract the hidden information only after generating the decrypted codeword's with hidden data.

Step4: The last bit encryption may change the sign of Level according to Levels and Corresponding Codeword's table. The extracted data bit is "0" when the decrypted codeword of Level belongs to code space C0. The extracted data bit is "1" when the decrypted codeword of Level belongs to code space C1.

Step5: According to the data hiding key, generate the same pseudo-random sequence P that was used in embedding process. In order to get the original additional information, the extracted bit sequence should be decrypted.

II. CONCLUSION

Because of the privacy-preserving requirements from cloud data management, encrypted media's data hiding is a new topic in the emerging world. To embed additional data in encrypted H.264/AVC bit stream is presented in this paper and can have video encryption, data embedding and data extraction. Even after encryption and data embedding, the algorithm should preserve the bit-rate correctly. As it is directly performed in the compressed and encrypted domain, it is simple to implement. Using codeword substitution, the content owner should embed the additional data into the encrypted bit stream without the knowledge of original video content. This can preserve the confidentiality of the content completely since data hiding is completed entirely in the encrypted domain. Data extraction can be done either in encrypted or decrypted domain and it can provide two different practical applications.

ACKNOWLEDGEMENT

We would like to extend our gratitude to the reference authors, as well as reviewer of our paper.

REFERENCES

[1] W. J. Lu, A. Varna, and M. Wu, "Secure video processing: Problems and challenges," in Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing, Prague, Czech Republic, May 2011, pp. 5856–5859.
[2] B. Zhao, W. D. Kou, and H. Li, "Effective watermarking scheme in the encrypted domain for buyer-seller watermarking protocol," Inf. Sci., vol. 180, no. 23, pp. 4672–4684, 2010.
[3] P. J. Zheng and J. W. Huang, "Walsh-Hadamard transform in the homomorphic encrypted domain and its application in image watermarking," in Proc. 14th Inf. Hiding Conf., Berkeley, CA, USA, 2012, pp. 1–15.
[4] W. Puech, M. Chaumont, and O. Strauss, "A reversible data hiding method for encrypted images," Proc. SPIE, vol. 6819, pp. 68191E-1–68191E-9, Jan. 2008.
[5] X. P. Zhang, "Reversible data hiding in encrypted image,"IEEE Signal Process. Lett. vol. 18, no. 4, pp. 255–258, Apr. 2011.
[6] W. Hong, T. S. Chen, and H. Y. Wu, "An improved reversible data hiding in encrypted images using side match," IEEE Signal Process. Lett. vol. 19, no. 4, pp. 199–202, Apr. 2012.