www.ijiarec.com

ISSN:2348-2079

Volume-6 Issue-2

International Journal of Intellectual Advancements and Research in Engineering Computations

Efficient Steganography in Encoded Video Streams Using Motion Vector Difference

M.Shankar[1], M.Usha[2], M.Ranjitha[2]

[1]Assistant Professor, Department of Information Technology,

[2]final year, Department of Information Technology,

M.P.Nachimuthu M.Jaganathan Engineering College

Email: ushamano2030@gmail.com

ABSTRACT : Generally, digital video sometimes are stored and processed in an encrypted format to maintain privacy and security. For the purpose of content notation, it is necessary to perform data hiding in these encrypted videos. In this way, data hiding in encrypted domain without decryption preserves the confidentiality of the content. In addition, it is more efficient without decryption followed by data hiding and re-encryption. This project proposes a novel scheme of data hiding directly in the encrypted version of AVI video stream, which includes the following three parts, i.e.,AVI video encryption, data embedding, and data extraction. By analyzing the property of AVI codec and the codeword substitution of motion vector differences are encrypted with stream ciphers. Then, a data hider may embed additional data in the encrypted domain by using codeword substitution technique, without knowing the original video content. In order to adapt to different application scenarios, data extraction can be done either in the encrypted domain or in the decrypted domain. Furthermore, video file size is strictly preserved even after encryption and data embedding.

Index Terms: Data hiding, encrypted domain, H.264/AVC, codeword substituting.

I. INTRODUCTION

CLOUD computing has become an important technology trend, which can provide highly efficient computation and large-scale storage solution for video data. Given that cloud services may attract more attacks and are vulnerable to untrustworthy system administrators, it is desired that the video content is accessible in encrypted form. The capability of performing data hiding directly in encrypted H.264/AVC video streams would avoid the leakage of video content, which can help address the security and privacy concerns with cloud computing. For example, a cloud server can embed the additional information (e.g., video notation, or authentication data) into an encrypted version of an H.264/AVC video by using data hiding technique. With the hidden information, the server can manage the video or verify its integrity without knowing the original content, and thus the security and privacy can be protected. In addition to cloud computing, this technology can also be applied to other prominent application scenarios. For example, when medical videos or surveillance videos have been encrypted for protecting the privacy of the people, a database manager may embed the personal information into the corresponding encrypted videos to provide the data management capabilities in the encrypted domain.

II. . PROPOSED SCHEME

In this section, a novel scheme of data hiding in the encrypted version of H.264/AVC videos is presented, which includes three parts, i.e., H.264/AVC video encryption, data embedding and data extraction. The content owner encrypts the original H.264/AVC video stream using standard stream ciphers with encryption keys to produce an encrypted video stream. Then, the data-hider (e.g., a cloud server) can embed the additional data into the encrypted video stream by using codeword substituting method, without knowing the original video content. At the receiver end, the hidden data extraction can be accomplished either in encrypted or in decrypted version. The diagram of the proposed framework is shown in Fig. 1, where the encryption and data embedding are depicted in part (a), and the data extraction and video decryption are shown in part (b).

A.Encryption of H.264/AVC Video Stream Video encryption often requires that the scheme be time efficient to meet the requirement of real time and format compliance. It is not practical to encrypt the whole compressed video bit stream like what the traditional ciphers do because of the following two constraints, i.e., format compliance and computational cost. Alternatively, only a fraction of video data is encrypted to improve the efficiency while still achieving adequate security. The key issue is then how to select the sensitive data to encrypt. According to the analysis given in [13], it is reasonable to encrypt both spatial information (IPM and residual data) and motion information (MVD) during H.264/AVC encoding.

In this paper, an H.264/AVC video encryption scheme with good performance including security, efficiency, and format compliance is proposed. By analyzing the property of H.264/AVC codec, three sensitive parts (i.e., IPMs, MVDs, and residual coefficients) are encrypted with stream ciphers. The proposed encryption algorithm is performed not during H.264/AVC encoding but in the H.264/AVC compressed domain. In this case, the bit stream will be modified directly. Selective encryption in the H.264/AVC compressed domain has been already presented on context-adaptive variable length coding (CAVLC) and context-adaptive binary arithmetic coding (CABAC) .In this paper, we have improved and enhanced the previous proposed approach by encrypting more syntax elements. We encrypt the codewords of IPMs, the codewords of MVDs, and the codewords of residual coefficients. The encrypted bitstream is still H.264/AVC compliant and can be decoded by any standard-compliant H.264/AVC decoder, but the encrypted video data is treated completely different compared to plaintext video data. In fact, performing the format-compliant encryption directly on the compressed bitstream is extremely complicated as the internal states of the encoder have to be preserved, otherwise the remaining data is interpreted falsely which may easily lead to format violations.

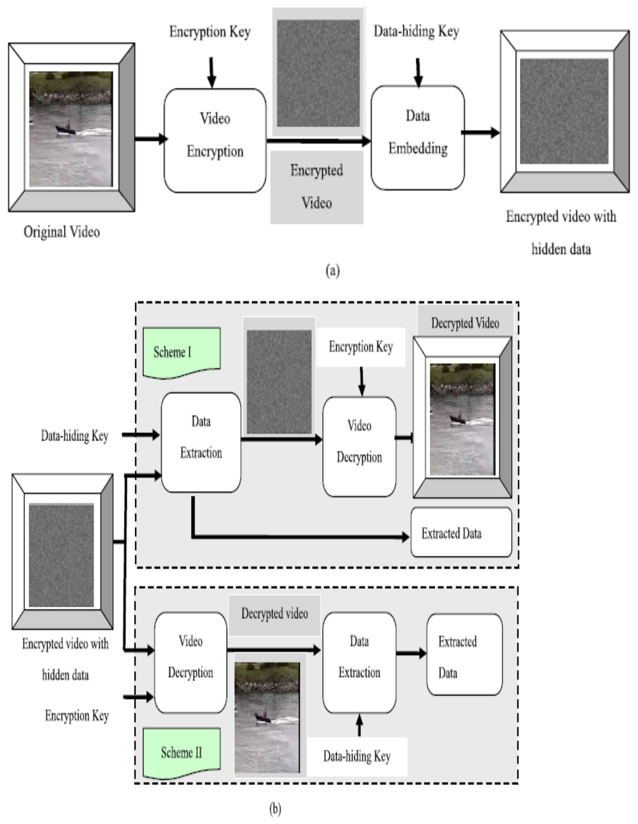


Fig. 1. Diagram of proposed scheme. (a) Video encryption and data embedding at the sender end. (b) Data extraction and video display at the receiver end in two scenarios.

***1)Intra-Prediction Mode (IPM) Encryption*:** According to AVI standard, the following four types of intra coding are supported, which are denoted as Intra_4 × 4, Intra_16× 16, Intra_chroma, and I_PCM. Here, IPMs in the Intra_4 × 4 and Intra_16 × 16 blocks are chosen to encrypt. Four intra prediction modes (IPMs) are available in the Intra_16 × 16. The IPM for Intra_16 × 16 block is specified in the mb_type (macroblock type) field which also specifies other parameters about this block such as coded block pattern (CBP).

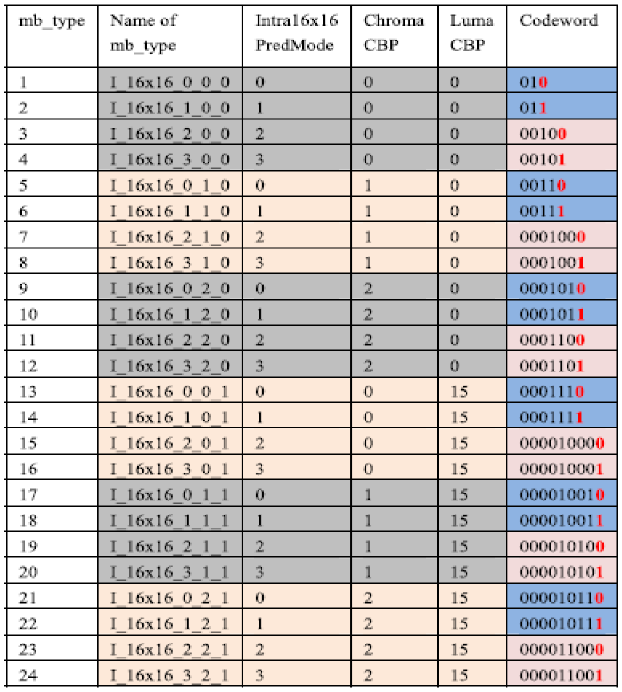
***2) Motion Vector Difference (MVD) Encryption**:* In order to protect both texture information and motion information, not only the IPMs but also the motion vectors should be encrypted. In AVI , motion vector prediction is further performed on the motion vectors, which yields MVD. In AVI baseline profile, Exp-Golomb entropy coding is used to encode MVD. The codeword of Exp-Golomb is constructed as[M zeros] [I N FO], where I N FO is an M-bit field carrying information.

3) Residual Data Encryption: In order to keep high security, another type of sensitive data, i.e., the residual data in both I-frames and P-frames should be encrypted. In AVI baseline profile, CAVLC entropy coding is used to encode the quantized coefficients of a residual block. Each CAVLC codeword can be expressed as the following format:

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

TABLE I

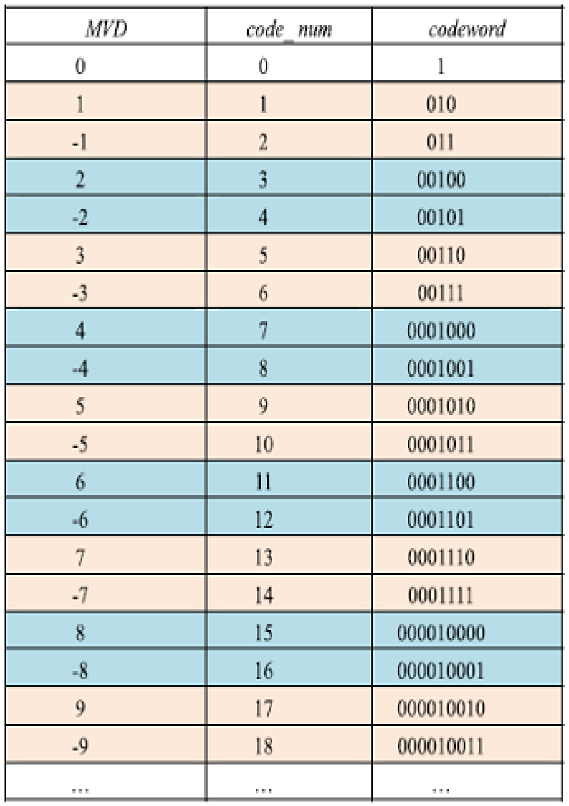
MACROBLOCK TYPES FOR I SLICES AND VARIABLE LENGTH OF CODEWORD IN H.264/AVC [17]



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
21	I_16x16_0_2_1	0	2	15	000010110
22	I_16x16_1_2_1	1	2	15	000010111
23	I_16x16_2_2_1	2	2	15	000011000
24	I_16x16_3_2_1	3	2	15	000011001

TABLE II

MVDs AND CORRESPONDING EXP-GOLOMB CODEWORDS



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
...	...	...

B. Data Embedding: In the encrypted bitstream of AVI , the proposed data embedding is accomplished by substituting eligible codewords of Levels. Since the sign of Levels are encrypted, data hiding should not affect the sign of Levels. Besides, the codewords substitution should satisfy the following three limitations.

First, the bitstream after codeword substituting must remain syntax compliance so that it can be decoded by standard decoder.

Second, to keep the bit-rate unchanged, the substituted codeword should have the same size as the original codeword.

Third, data hiding does cause visual degradation but the impact should be kept to minimum. That is, the embedded data after video decryption has to be invisible to a human observer.

So the value of Level corresponding to the substituted codeword should keep close to the value of Level corresponding to the original codeword. In addition, the codewords of Levels within P-frames are used for data hiding, while the codewords of Levels in I-frames are remained unchanged. Because I-frame is the first frame in a group of pictures (GOPs), the error occurred in I-frame will be propagated to subsequent P-frames.

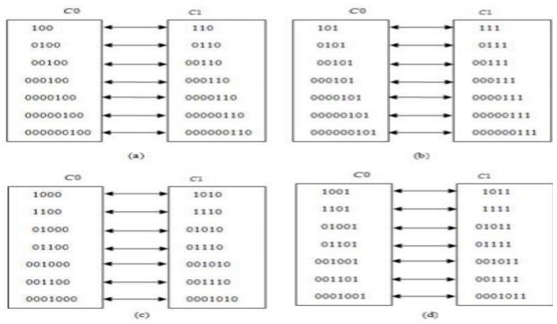


Fig. 2. CAVLC codeword mapping. (a) suffix Length = 2& Level > 0. (b) suffix Length = 2& Level > 0. (c) suffix Length = 3& Level > 0. (d) suffix Length = 3& Level < 0

C. *Data Extraction:*

Scheme I: Encrypted Domain Extraction

To protect privacy, a database manager (e.g., cloud server) may only get access to the data hiding key and have to manipulate data in encrypted domain.

In encrypted domain, encrypted video with hidden data is directly sent to the data extraction module, and the extraction process is given as follows.

Step1: The codewords of Levels are firstly identified by parsing the encrypted bitstream.

Step2: If the codeword belongs to codespace C0, the extracted data bit is “0”. If the codeword belongs to codespace C1, the extracted data bit is “1”.

Step3: According to the data hiding key, the same chaotic pseudo-random sequence P that was used in the embedding process can be generated. Then the extracted bit sequence could be decrypted by using P to get the original additional information. Since the whole process is entirely operated in encrypted domain, it effectively avoids the leakage of original video content.

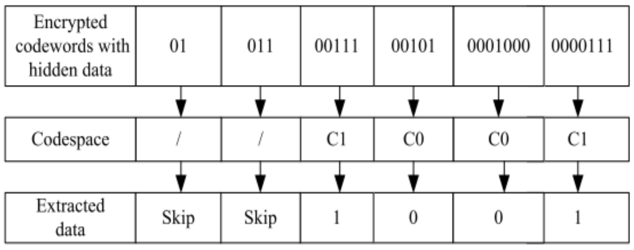
Scheme II: Decrypted Domain Extraction.

In scheme I, both embedding and extraction of the data are performed in encrypted domain. However, in some cases, users want to decrypt the video first and extract the hidden data from the decrypted video. For example, an authorized user, which owned the encryption key, received the encrypted video with hidden data. The received video can be decrypted using the encryption key. That is, the decrypted video still includes the hidden data, which can be used to trace the source of the data. Data extraction in decrypted domain is suitable for this case.

Step1: Generate encryption streams with the encryption keys as given in encryption process.

Step2:The codewords of IPMs, MVDs, Sign_of_TrailingOnes and Levels are identified by parsing the encrypted bitstream.

Step3: The decryption process is identical to the encryption process, since XOR operation is symmetric. The encrypted codewords can be decrypted by performing XOR operation with generated encryption streams, and then two XOR operations cancel each other out, which renders the original plaintext. Since the encryption streams depend on the encryption keys, the decryption is possible only for the authorized users. After generating the decrypted codewords with hidden data, the content owner can further extract the hidden information.



Data extraction in encrypted domain

Step4: The last bit encryption may change the sign of Level. Encrypted codeword and the original codeword are still in the same codespaces. If the decrypted codeword of Level belongs to codespace C0, the extracted data bit is “0”. If the decrypted codeword of Level belongs to codespace C1, the extracted data bit is “1”.

Step5: Generate the same pseudo-random sequence P that was used in embedding process according to the data hiding key. The extracted bit sequence should be decrypted to get the original additional information.

III. ARCHITECTURAL DIAGRAM:

The process of data hiding method consists of following three main procedures such as Image encryption, data embedding and data extraction/image recovery. The separable scheme the content owner encrypts the original uncompressed image using an encryption key to produce an encrypted image. Then using a data-hiding key the data-hider compresses the Least Significant Bits (LSB) of the encrypted image to create some space to accommodate the additional data.

At the receiver side the data embedded can be easily retrieved from the encrypted image containing additional data according to the data-hiding key. As the data embedding only affects the LSB a decryption with the encryption key may result in an image similar to the original version. With an encrypted image containing additional data which is hidden case one is when the receiver has only the data-hiding key, he is able to extract the additional data even if he does not know the image content

IV. CONCLUSION

Data hiding in encrypted media is a new topic that has started to draw attention because of the privacy-preserving requirements from cloud data management. In this paper, an algorithm to embed additional data in encrypted H.264/AVC bitstream is presented, which consists of video encryption, data embedding and data extraction phases. The algorithm can preserve the bit-rate exactly even after encryption and data embedding, and is simple to implement as it is directlyperformed in the compressed and encrypted domain, i.e., it does not require decrypting or partial decompression of the video stream thus making it ideal for real-time video applications. The data-hider can embed additional data into the encrypted bitstream using codeword substituting, even though he does not know the original video content. Since data hiding is completed entirely in the encrypted domain, our method can preserve the confidentiality of the content completely. With an encrypted video containing hidden data, data extraction can be carried out either in encrypted or decrypted domain, which provides two different practical applications. Another advantage is that it is fully compliant with the H.264/AVC syntax. Experimental results have shown that the proposed encryption and data embedding scheme can preserve file-size, whereas the degradation in video quality caused by data hiding is quite small.

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.