# Data Hiding in Encrypted H.264/AVC Video Streams by Codeword Substitution

Dawen Xu, Rangding Wang, and Yun Q. Shi, *Fellow, IEEE*

***Abstract*— Digital video sometimes needs to be stored and processed in an encrypted format to maintain security and privacy. For the purpose of content notation and/or tampering detection, 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. In this paper, a novel scheme of data hiding directly in the encrypted version of H.264/AVC video stream is proposed, which includes the following three parts, i.e., H.264/AVC video encryption, data embedding, and data extraction. By analyzing the property of H.264/AVC codec, the codewords of intraprediction modes, the codewords of motion vector differences, and the codewords of residual coefficients 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. Experimental results have demonstrated the feasibility and efficiency of the proposed scheme.**

***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 [1]. 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.

Till now, few successful data hiding schemes in the encrypted domain have been reported in the open literature. In [2], a watermarking scheme in the encrypted domain using Paillier cryptosystem is proposed based on the security requirements of buyer-seller watermarking protocols. A Walsh-Hadamard transform based image watermarking algorithm in the encrypted domain using Paillier cryptosystem is presented in [3]. However, due to the constraints of the Paillier cryptosystem, the encryption of an original image results in a high overhead in storage and computation. Note that, several investigations on reversible data hiding in encrypted images are reported in [4]–[8] recently. The encryption is performed by using bit-XOR (exclusive-OR) operation. In these methods, however, the host image is in an uncompressed format. In [9], a robust watermarking algorithm is proposed to embed watermark into compressed and encrypted JPEG2000 images.

As said the above mentioned works have been focused on image. With the increasing demands of providing video data security and privacy protection, data hiding in encrypted H.264/AVC videos will undoubtedly become popular in the near future. Obviously, due to the constraint of the underlying encryption, it is very difficult and sometimes impossible to transplant the existing data hiding algorithms to the encrypted domain. To the best of our knowledge, there has been no report on the implementation of data hiding in encrypted H.264/AVC video streams. Only few joint data-hiding and encryption approaches that focus on video have been proposed. For example, in [10], during H.264/AVC compression, the intra-prediction mode (IPM), motion vector difference (MVD) and DCT coefficients' signs are encrypted, while DCT coefficients' amplitudes are watermarked adaptively. In [11], a combined scheme of encryption and watermarking is presented,

Manuscript received November 12, 2013; accepted January 21, 2014. Date of publication January 27, 2014; date of current version March 4, 2014. This work was supported in part by the Natural Science Foundation of China under Grants 61301247 and 61170137, in part by Zhejiang Provincial Natural Science Foundation of China under Grant LY13F020013, and in part by Ningbo Natural Science Foundation under Grant 2013A610059. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Adnan M. Alattar.

D. Xu is with the School of Electronics and Information Engineering, Ningbo University of Technology, Ningbo 315016, China (e-mail: dawenxu@126.com).

R. Wang is with the CKC Software Laboratory, Ningbo University, Ningbo 315211, China (e-mail: wangrangding@nbu.edu.cn).

Y. Q. Shi is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102-1982 USA (e-mail: shi@njit.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2014.2302899

which can provide the access right as well as the authentication of video content simultaneously. The IPMs of $4 \times 4$ luminance block, the sign bits of texture, and the sign bits of MVDs are encrypted, while IPM is used for watermarking. However, the watermarked bitstream is not fully format-compliant as a result a standard decoder may crash since it cannot parse a watermarked bitstream. Concretely, the value "$-2$" of IPM does not exist in the actual standard. In summary, in the existing related technologies [10]–[11], encryption and data embedding are implemented almost simultaneously during H.264/AVC compression process. However, to meet the aforementioned application requirements, it's necessary to perform data hiding directly in the encrypted domain. In addition, the approaches in [10] and [11] do not operate on the compressed bitstream. That is, encryption and watermark embedding are accomplished in the encoding process, while decryption and watermark detection are completed in the decoding process The compression/decompression cycle is time-consuming and hampers real-time implementation. Besides, encryption and watermark embedding would lead to increasing the bit-rate of H.264/AVC bitstream.

Therefore, it becomes highly desirable to develop data hiding algorithms that work entirely on encoded bitstream in the encrypted domain However, there are some significant challenges for data hiding directly in compressed and encrypted bitstream. The first challenge is to determine where and how the bitstream can be modified so that the encrypted bitstream with hidden data is still a compliant compressed bitstream. The second challenge is to insure that decrypted videos containing hidden data can still appear to be of high visual fidelity. The third challenge is to maintain the file size after encryption and data hiding, which requires that the impact on compression gain is minimal. The fourth challenge is that the hidden data can be extracted either from the encrypted video stream or from the decrypted video stream, which is much more applicable in practical applications.

Based on the analysis given above, we propose a novel scheme to embed secret data directly in compressed and then encrypted H.264/AVC bitstream. Firstly, by analyzing the property of H.264/AVC codec, the codewords of IPMs, the codewords of MVDs, and the codewords of residual coefficients are encrypted with a stream cipher. The encryption algorithm is combined with the Exp-Golomb entropy coding and Context-adaptive variable-length coding (*CAVLC*) [12], which keeps the codeword length unchanged. Then, data hiding in the encrypted domain is performed based on a novel codeword substituting scheme. In contrast to the existing technologies [10]–[11] discussed above, the proposed scheme can achieve excellent performance in the following three different prospects.

- The data hiding is performed directly in encrypted H.264/AVC video bitstream.
- The scheme can ensure both the format compliance and the strict file size preservation.
- The scheme can be applied to two different application scenarios by extracting the hidden data either from the encrypted video stream or from the decrypted video stream.

The remainder of the paper is organized as follows. In Section II, we describe the proposed scheme, which includes three parts, i.e., H.264/AVC video encryption, data embedding and data extraction. Experimental results are presented in Section III. Discussion is shown in Section IV. Finally in Section V, conclusion is drawn.

## 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 bitstream 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. Compared with [13], the proposed encryption algorithm is performed not during H.264/AVC encoding but in the H.264/AVC compressed domain. In this case, the bitstream will be modified directly. Selective encryption in the H.264/AVC compressed domain has been already presented on context-adaptive variable length coding (*CAVLC*) [14] and context-adaptive binary arithmetic coding (*CABAC*) [15]. 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

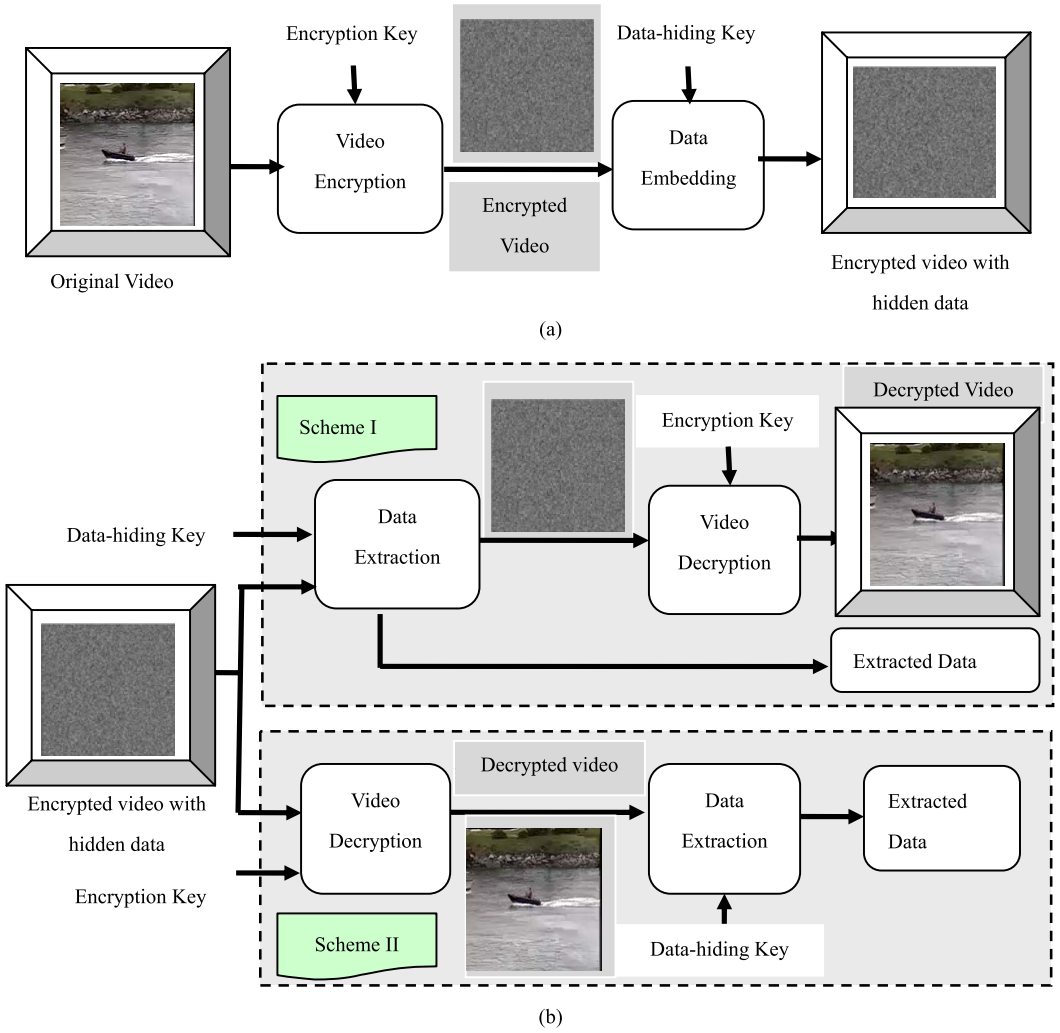


Fig. 1. Diagram of proposed scheme. (a) Video encryption and data embedding at the sender end. (b) Data extraction and video display at the receiver end in two scenarios.

encryption directly on the compressed bitstream is extremely complicated as the internal states of the encoder have to be preserved [16], otherwise the remaining data is interpreted falsely which may easily lead to format violations.

*1) Intra-Prediction Mode (IPM) Encryption:* According to H.264/AVC 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 [12]. 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). Table I is the list of mb_type values with their meanings which are taken from the standard [17]. In H.264/AVC baseline profile, the mb_type is encoded with the Exp-Golomb code. To maintain standard-compliance of the encrypted bitstream, we can encrypt the codeword of an IPM without modifying the CBP. In addition, to keep the codeword's length unchanged, the encrypted codeword should have the same size as the original codeword. It can be observed that the combination of CBP is the same in every four lines, and the codewords have the same length in every two consecutive lines, as shown in Table I. For example, the codewords corresponding to "3" and "4" are "00100" and "00101", respectively, which have the same length and the same value of CBP. Thus for Intra_16 × 16 block, the IPM encryption is performed by applying a bitwise XOR operation between the last bit of the codewords and a bit of the pseudorandom sequence to keep the value of CBP and the length of codeword unchanged [18]. The pseudorandom sequence is generated via a standard secure cipher (e.g., RC4) determined by an encryption key *E_Key*1.

In H.264/AVC, each Intra_4 × 4 luminance block is predicted from its spatially neighboring samples. Specifically, H.264/AVC offers nine prediction modes (0-8) for Intra_4 × 4 luminance blocks. The choice of prediction mode for each Intra_4 × 4 luminance block must be signalled to the decoder and this could potentially require a large number of bits. To efficiently compress the prediction-mode data, the predictive coding technique is applied to signal prediction modes. For each currently considered block $E$, the most probable

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 |

mode ($MPM_E$) is defined as the smaller of the prediction modes of the spatially adjacent upper block $A$ and left block $B$ [19]. If either of these adjacent blocks is not available, the corresponding value is set to 2, indicating "DC" prediction mode.

The prediction mode of the currently considered block $E$ is denoted as $Mode_E$. If $Mode_E$ is equal to$MPM_E$, only one bit is needed to signal the prediction mode. When $Mode_E$ and $MPM_E$ are different, the codeword is composed of one flag bit "0"and three bits fixed-length code [19].

For Intra_4 $\times$ 4 block, if $Mode_E$ is equal to $MPM_E$, the codeword is kept unchanged. Otherwise, three bits fixed-length code (denoted as $X$) [13] in each codeword is encrypted with a pseudorandom sequence which is generated via a standard secure cipher (e.g., RC4) determined by an encryption key $E\_Key2$. Bitwise XOR operation is still utilized as the encryption scheme.

From what described above, it is obvious that the length of the encrypted codeword is the same as the original one. For the format compliance in the decoding process, the encrypted IPMs of blocks in the first row and/or in the first column should have the decodable value, since not all modes are available along the top and the left borders of each frame due to the lack of neighbors. In our scheme, if the IPM after encryption is not available for a border block, then the IPM encryption of this block will be skipped. This further indicates that IPM encryption is not secure enough in some specific locations and should be used in combination with other encrypting method. In summary, IPM encryption implies changing the actual mode to another one without violating the semantics and bitstream compliance.

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 |
| … | … | … |

*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 H.264/AVC, motion vector prediction is further performed on the motion vectors, which yields MVD. In H.264/AVC baseline profile, Exp-Golomb entropy coding [19] is used to encode MVD. The codeword of Exp-Golomb is constructed as$[M\ zeros]$ [1] $[INFO]$, where $INFO$ is an $M$-bit field carrying information.

Table II shows the values of MVDs and corresponding Exp-Golomb codewords. The last bit of the codeword is encrypted by applying the bitwise XOR operation with a standard stream cipher, which is determined by an encryption key $E\_Key3$. According to Table II, the last bit encryption may change the sign of MVD, but does not affect the length of the codeword and satisfies the format compliance [10]. That is, the resulting ciphertexts are still valid Exp-Golomb codes. For example, the codewords corresponding to "2" and "−2" are "00100" and "00101", respectively, which have the same length. It should be noted that when the value of MVD is equal to 0, its corresponding codeword "1" keeps unchanged during the encryption process.

*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 this section, a novel method for encrypting the residual data based on the characteristics of codeword is presented in detail.

In H.264/AVC baseline profile, *CAVLC* entropy coding is used to encode the quantized coefficients of a residual block [19]. Each *CAVLC* codeword can be expressed as the following format:

$$\{Coeff\_token, Sign\_of\_TrailingOnes, Level, Total\_zeros, Run\_before\}$$

The specific function of each syntax element is described in [19]. To keep the bitstream compliant, not all syntax elements can be modified during encryption process. For example,$Coeff\_token$, $Total\_zeros$, and $Run\_before$ should remain unchanged [14]. Therefore, residual data encryption can be accomplished by modifying the codewords of *Sign_of_TrailingOnes* and *Level*.

The *Sign_of_TrailingOnes* is encoded with a single bit. Bit "0" is assigned for $+1$ and bit "1"is assigned for $-1$. The codeword of *Sign_of_TrailingOnes* is encrypted by applying the bitwise XOR operation with a standard stream cipher, which is determined by an encryption key $E\_Key4$.

The codeword for each *Level* is made up of a prefix (*level_prefix*) and a suffix (*level_suffix*) as

$$Level\ codeword = [level\_prefix], [level\_suffix]$$

Table III shows *Levels* with different *suffixLength* and corresponding *codewords*. The last bit of the codeword is encrypted by applying the bitwise XOR operation with a standard stream cipher, which is determined by an encryption key $E\_Key5$. According to Table III, the last bit encryption may change the sign of *Levels*, but does not affect the length of the codeword and satisfies the format compliance. For example, when *suffixLength* is equal to 1, the codewords corresponding to "2" and "-2" are "010" and "011", respectively, which have the same length. It should be noted that when *suffixLength* is equal to 0, the codewords should keep unchanged during the encryption process.

TABLE III
LEVELS AND CORRESPONDING CODEWORDS

| *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 |
| | 12 | 00000110 | -12 | 00000111 |
| | 13 | 000000100 | -13 | 000000101 |
| | 14 | 000000110 | -14 | 000000111 |
| 3 | 1 | 1000 | -1 | 1001 |
| | 2 | 1010 | -2 | 1011 |
| | 3 | 1100 | -3 | 1101 |
| | 4 | 1110 | -4 | 1111 |
| | 5 | 01000 | -5 | 01001 |
| | 6 | 01010 | -6 | 01011 |
| | 7 | 01100 | -7 | 01101 |
| | 8 | 01110 | -8 | 01111 |
| | 9 | 001000 | -9 | 001001 |
| | 10 | 001010 | -10 | 001011 |
| | 11 | 001100 | -11 | 001101 |
| | 12 | 001110 | -12 | 001111 |
| | 13 | 0001000 | -13 | 0001001 |
| | 14 | 0001010 | -14 | 0001011 |

### *B. Data Embedding*

Although few methods have been proposed to embed data into H.264/AVC bitstream directly [20]–[21], however, these methods cannot be implemented in the encrypted domain. In the encrypted bitstream of H.264/AVC, the proposed data embedding is accomplished by substituting eligible codewords of *Levels* in Table III. 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.

According to the analysis given above, we can see that there are no corresponding substituted codewords when *suffixLength* is equal to 0 or 1, as shown in Table III. When *suffixLength* is equal to 0, we cannot find a pair of codewords with the same size. When *suffixLength* is equal to 1, one codeword also cannot be substituted by another codeword with the same size, since this substitution would change the sign of *Level*. Then the codewords of *Levels* which *suffixLength* is 2 or 3 would be

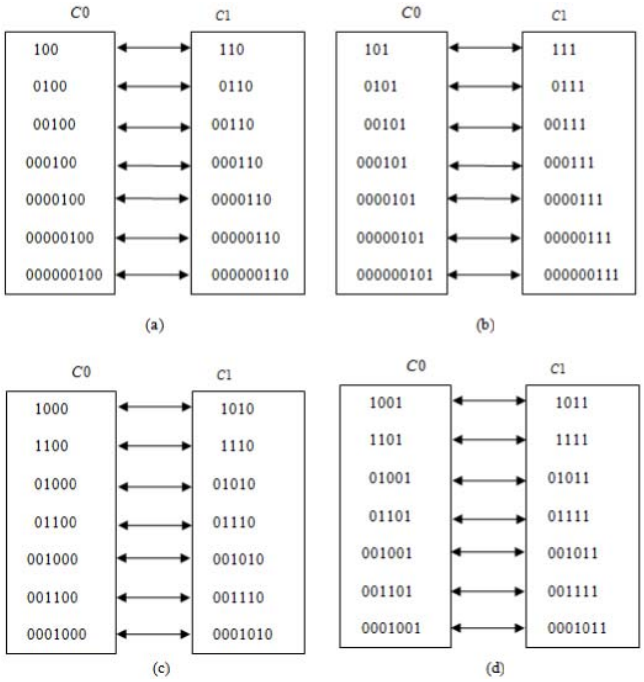


Fig. 2. CAVLC codeword mapping. (a) $suffix\ Length = 2\&\ Level > 0$. (b) $suffix\ Length = 2\&\ Level > 0$. (c) $suffix\ Length = 3\&\ Level > 0$. (d) $suffix\ Length = 3\&\ Level < 0$.

divided into two opposite codespaces denoted as $C0$and $C1$ as shown in Fig. 2. The codewords assigned in $C0$and $C1$ are associated with binary hidden information "0" and "1".

Suppose the additional data that we want to embed is a binary sequence denoted as $B = \{b(i)|i = 1, 2, \cdots, L, b(i) \in \{0, 1\}\}$. Data hiding is performed directly in encrypted bitstream through the following steps.

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

Step2. The codewords of *Levels* are obtained by parsing the encrypted H.264/AVC bitstream.

Step3. If current codeword belongs to codespaces $C0$or$C1$, the to-be-embedded data bit can be embedded by codeword substituting. Otherwise, the codeword is left unchanged. The detailed procedure of codeword substituting for data hiding is shown in Fig. 3. For example, when *Level* is positive 1 and its *sufflxLength* is 3, then its corresponding codeword is "1000" which belongs to *C0* as shown in Fig. 2(c). If the data bit "1" needs to be embedded, the codeword "1000" should be replaced with "1010". Otherwise, if the data bit "0" needs to embedded, the codeword "1000" will keep unchanged.

Step4. Choose the next codeword and then go to Step3 for data hiding. If there are no more data bits to be embedded, the embedding process is stopped.

Suppose the to-be-embedded data is "1001", the *CAVLC* codeword of *Level* parsing from H.264/AVC bitstream is

```
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. 3. The procedure of codeword mapping.

"01 010 00100 00100 0001011 0000100" and the encryption stream is "10111", an example of data embedding based on codeword mapping is shown in Fig. 4(a).

### C. Data Extraction

In this scheme, the hidden data can be extracted either in encrypted or decrypted domain, as shown in Fig. 1(b). Data extraction process is fast and simple. We will first discuss the extraction in encrypted domain followed by decrypted domain.

*1) 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. Data extraction in encrypted domain guarantees the feasibility of our scheme in this case.

In encrypted domain, as shown in Fig. 1(b), 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.

An example of data extraction in encrypted domain is shown in Fig. 4(b).

*2) 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

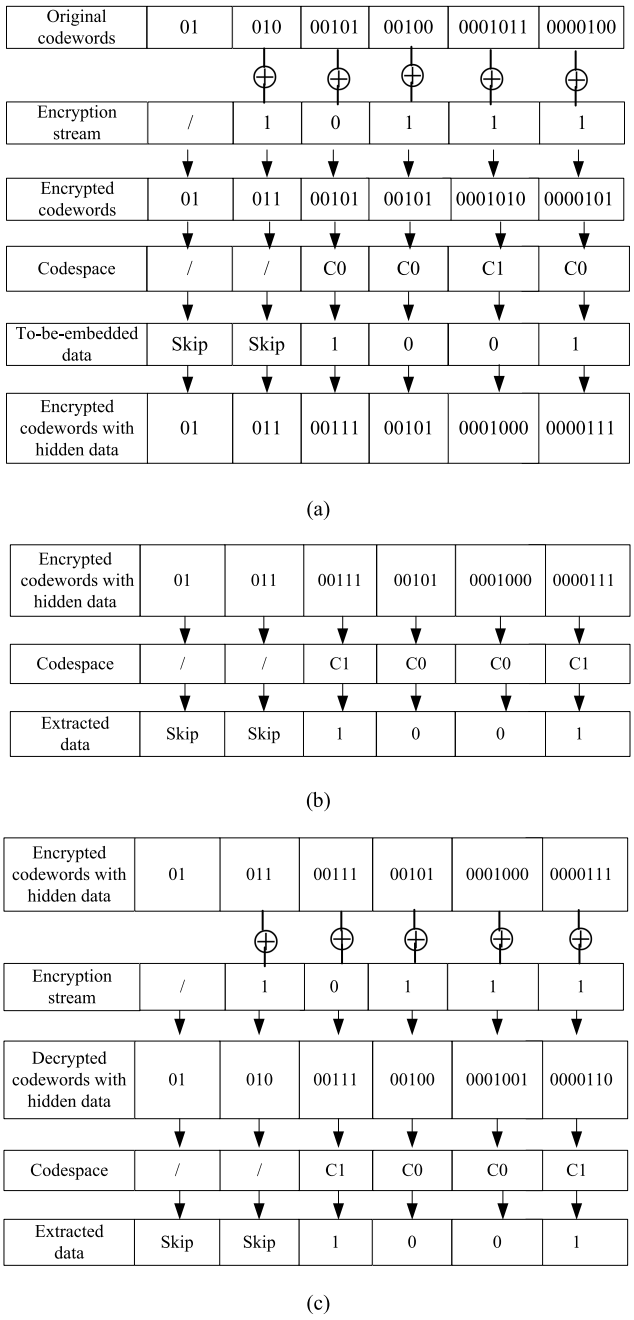


Fig. 4. An example of data embedding and extraction. (a) Data embedding. (b) Data extraction in encrypted domain. (c) Data extraction in decrypted domain.

case. As shown in Fig. 1(b), the received encrypted video with hidden data is first pass through the decryption module. The whole process of decryption and data extraction is given as follows.

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.

Step4: According to Table III, the last bit encryption may change the sign of *Level*. However, as shown in Fig. 2, the 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.

An example of data extraction in decrypted domain is shown in Fig. 4(c).

## III. Experimental Results

The proposed data hiding scheme has been implemented in the H.264/AVC reference software version JM-12.2. Six well-known standard video sequences (i.e., *Stefan*, *Table*, *Tempete*, *Mobile*, *Hall*, *and News*) in QCIF format ($176 \times 144$) at the frame rate 30 frames/s are used for simulation. The first 100 frames in each video sequence are used in the experiments. The GOP (Group of Pictures) structure is "IPPPP: one I frame followed four P frames".

### A. Security of Encryption Algorithm

For the proposed video encryption scheme, the security includes both cryptographic security and perceptual security. Cryptographic security denotes the security against cryptographic attacks, which depends on the ciphers adopted by the scheme. In the proposed scheme, the secure stream cipher (e.g., RC4) is used to encrypt the bitstream, and chaotic pseudo-random sequence generated by logistic map is used to encrypt the additional data. They have been proved to be secure against cryptographic attacks. Perceptual security refers to whether the encrypted video is unintelligible or not. Generally, it depends on the encryption scheme's properties. For example, encrypting only IPM cannot keep secure enough, since the encrypted video is intelligible [10]. The proposed scheme encrypts IPM, MVD and residual coefficients, which keeps perceptual security of the encrypted video. The demonstration is shown in Figs. 5 and 6. An original frame from each video is depicted in Fig. 5, and their corresponding encrypted results are depicted in Fig. 6. Other frames have a similar effect of encryption. Due to space limitations, we do not list the results of all frames. It should be mentioned that not every video can be degraded to the same extent. The perceptual quality of high-motion videos with a complex textured background becomes much more scrambled after encryption than that of low-motion videos with a static background. The reason is that there are less residual coefficients and MVDs in low-motion

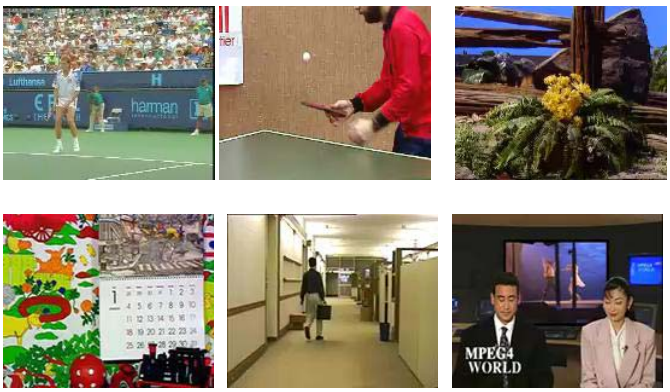


Fig. 5. Original video frames.

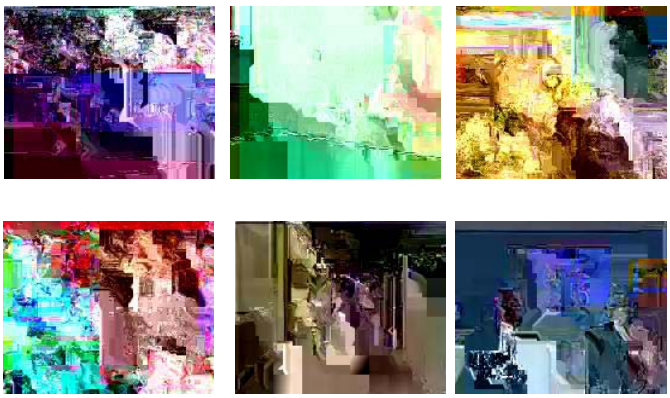

Fig. 6. Encrypted video frames.

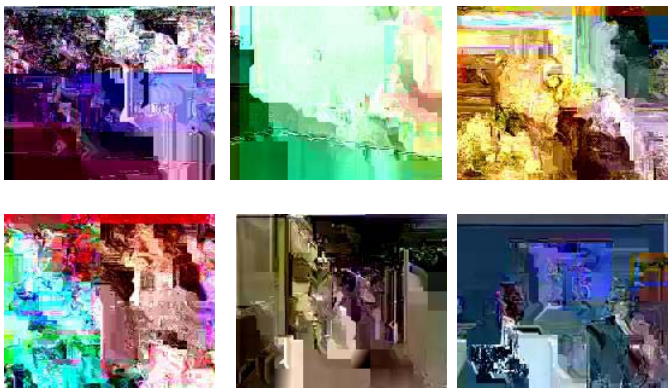

Fig. 7. Encrypted video frames with hidden data.

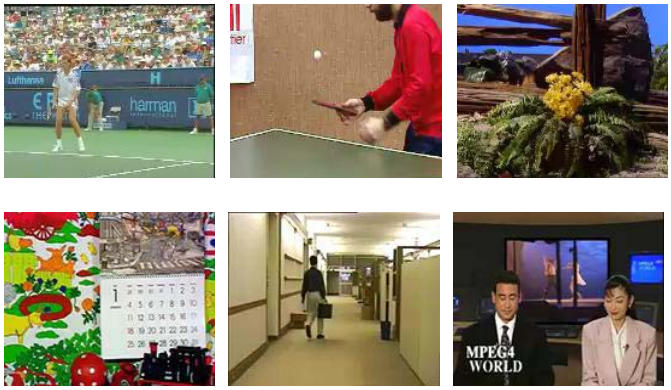


Fig. 8. Decrypted video frames with hidden data.

videos that are available for encryption. In general, scrambling performance of the described encryption system is more than adequate.

### *B. Visual Quality of Stego Video*

The encrypted video containing hidden data provided by the server should be decrypted by the authorized user. Therefore, the visual quality of the decrypted video containing hidden data is expected to be equivalent or very close to that of the original video. By modifying the compressed bitstream to embed additional data, the most important challenge is to maintain perceptual transparency, which refers to the modification of bitstream should not degrade the perceived content quality. In this paper, only the codewords of *Levels* within P-frames are modified for data hiding. Simulation results have demonstrated that we can embed the additional data with a large capacity into P-frames while preserving high visual quality. The encrypted and decrypted video frames with hidden data are shown in Figs 7 and 8 respectively. In the experiments, no visible artifacts have been observed in all of the decrypted video frames with hidden data.

Besides subjective observation, *PSNR* (Peak Signal to Noise Ratio), *SSIM* (Structural Similarity Index), and *VQM* (Video Quality Measurement) have been adopted to evaluate the perceptual quality [22]. *PSNR* is widely used objective video quality metric. However, it does not perfectly correlate with a perceived visual quality due to nonlinear behavior of human visual system. The *SSIM* index lies in the range between 0 and 1, where 1 indicates the reference image is identical than the target image. Since H.264/AVC is lossy compression, in order to better illustrate the data hiding on the video quality, the visual quality of non-stego video stream should be tested. The video sequence obtained by decompressing non-stego video stream is used as the target sequence, while the original uncompressed video sequence is used as the reference video sequence. Similarly, in order to test the visual quality of stego video stream, the video sequence obtained by encrypting, data hiding, decrypting, and decompressing process is used as the target sequence. That is, in this case, the target video contains hidden data. The *VQM* is another approach to measure video quality that correlates more with the human visual system. In general, the lower *VQM* value indicates higher perceptual video quality, and zero indicates excellent quality. The experimental results are shown in Table IV. As can be seen, a higher QP (quantization parameter) will result in lower video quality. The visual quality degradation of decrypted video containing hidden data is very low even for large payloads, i.e., it is generally hard to detect the degradation in video quality caused by data hiding.

### *C. Embedding Capacity*

Data hiding payload can be assessed in kilobits per second (kbits/s) [23]. The maximum payload capacity in each video

TABLE IV
EMBEDDING CAPACITY, PSNR, SSIM, AND VQM IN DIRECTLY DECRYPTED VIDEOS (CAVLC CODEWORD MAPPING)

| *Sequence* | *QP* | *Maximum Capacity* (kbits/s) | *PSNR* (dB) | | *SSIM* | | *VQM* | |
|---|---|---|---|---|---|---|---|---|
| | | | *Non-stego* | *Stego* | *Non-stego* | *Stego* | *Non-stego* | *Stego* |
| *Stefan* | 24 | 17.80 | 39.60 | 38.33 | 0.9833 | 0.9825 | 0.7385 | 0.7855 |
| | 28 | 3.63 | 35.84 | 35.50 | 0.9692 | 0.9688 | 1.0334 | 1.0586 |
| | 32 | 0.57 | 31.68 | 31.62 | 0.9447 | 0.9446 | 1.3834 | 1.3967 |
| *Table* | 24 | 8.27 | 38.44 | 37.91 | 0.9542 | 0.9537 | 0.6896 | 0.7193 |
| | 28 | 3.45 | 35.15 | 34.87 | 0.9091 | 0.9087 | 0.9246 | 0.9428 |
| | 32 | 0.82 | 32.31 | 32.17 | 0.8451 | 0.8449 | 1.1829 | 1.1970 |
| *Tempete* | 24 | 7.89 | 38.68 | 38.16 | 0.9855 | 0.9850 | 0.8586 | 0.8913 |
| | 28 | 1.13 | 34.83 | 34.74 | 0.9716 | 0.9714 | 1.2056 | 1.2210 |
| | 32 | 0.14 | 30.88 | 30.87 | 0.9448 | 0.9448 | 1.6372 | 1.6399 |
| *Mobile* | 24 | 1.81 | 38.44 | 38.32 | 0.9871 | 0.9871 | 0.8652 | 0.8724 |
| | 28 | 0.22 | 34.51 | 34.49 | 0.9741 | 0.9741 | 1.2552 | 1.2575 |
| | 32 | 0.01 | 30.63 | 30.63 | 0.9501 | 0.9501 | 1.7154 | 1.7154 |
| *Hall* | 24 | 0.60 | 40.33 | 40.26 | 0.9744 | 0.9743 | 0.6536 | 0.6573 |
| | 28 | 0.13 | 37.92 | 37.90 | 0.9658 | 0.9658 | 0.7848 | 0.7867 |
| | 32 | 0.02 | 34.98 | 34.98 | 0.9527 | 0.9527 | 0.9542 | 0.9552 |
| *News* | 24 | 0.50 | 40.82 | 40.75 | 0.9848 | 0.9848 | 0.5847 | 0.5872 |
| | 28 | 0.11 | 37.78 | 37.76 | 0.9727 | 0.9727 | 0.7745 | 0.7751 |
| | 32 | 0.02 | 34.57 | 34.56 | 0.9531 | 0.9531 | 1.0064 | 1.0065 |

encoded with different *QP* values is given in Table IV. Payload of the proposed scheme depends on type of video content and the *QP* values. The reason for this is that each video stream has a different number of qualified codewords. Embedding capacity is determined by the number of qualified codewords. Obviously, high motion sequence (e.g., *Stefan*, *Table*) has much larger number of qualified codewords. This is be caused by high motion sequence has more qualified *Levels*, since data hiding is only performed within P-frames. Specifically, payload decreases with increase in *QP* value. When QP value increases, the number of residual coefficients decreases, and then the qualified *Levels* will be less.

### D. Bit Rate Variation

To further evaluate the performance of the proposed scheme, bit rate variation *BR_var* caused by encryption and data hiding is also introduced.

$$BR\_var = \frac{BR\_em - BR\_orig}{BR\_orig} \times 100\%$$

where *BR_em* is the bit rate generated by encryption and data embedding encoder, and *BR_orig* is the bit rate generated by the original encoder. The bit rate of the encrypted and stego video remains unchanged. This is because the encryption and data hiding are all performed by replacing a suitable codeword to another codeword with the same length, as described in Section II (A) and (B). In summary, the encryption process and data hiding process do not affect compression efficiency of encoders, since compression efficiency is typically defined by the bit rate.

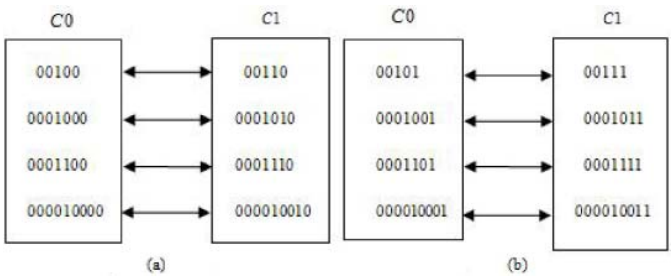


Fig. 9. *Exp-Golomb* codeword mapping. (a) $MVD > 0$. (b) $MVD < 0$.

## IV. DISCUSSION

In addition to the codewords of *Level*s, the Exp-Golomb codewords of *MVD*s can also be used for data hiding. From Table II, we can see that there are no corresponding substituted codewords when *MVD* is equal to 0, because we cannot find a pair of codeword with the same size. In addition, the codewords "010" and "011"should not be modified during data hiding, since they are encrypted codeword pair. Then the rested codewords would be divided into two opposite codespaces denoted as $C0$and $C1$as shown in Fig. 9. The codewords assigned in $C0$and $C1$ are associated with binary hidden information "0" and "1".

Data hiding and extraction procedure are the same as the previously described in Section II (B) and (C), respectively. The experiment results are shown in Table V. According to Table V, for high motion sequences (such as *Stefan*, *Table*) and high texture sequences (such as *Tempete* and *Mobile*), the degradation in video quality caused by *MVD*'s Exp-Golomb codeword substituting is more serious than the previous *Level's*

TABLE V
EMBEDDING CAPACITY, PSNR, SSIM, AND VQM IN DIRECTLY DECRYPTED VIDEOS (EXP-GOLOMB CODEWORD MAPPING)

| *Sequence* | *QP* | *Maximum Capacity* (kbits/s) | *PSNR* (dB) | | *SSIM* | | *VQM* | |
|---|---|---|---|---|---|---|---|---|
| | | | *Original* | *Stegoed* | *Original* | *Stegoed* | *Original* | *Stegoed* |
| *Stefan* | 24 | 2.04 | 39.60 | 32.10 | 0.9833 | 0.9544 | 0.7385 | 1.1360 |
| | 28 | 1.88 | 35.84 | 30.32 | 0.9692 | 0.9351 | 1.0334 | 1.4417 |
| | 32 | 1.71 | 31.68 | 28.96 | 0.9447 | 0.9236 | 1.3834 | 1.6630 |
| *Table* | 24 | 3.87 | 38.44 | 31.34 | 0.9542 | 0.9057 | 0.6896 | 1.1272 |
| | 28 | 2.94 | 35.15 | 29.66 | 0.9091 | 0.8632 | 0.9246 | 1.3144 |
| | 32 | 1.97 | 32.31 | 28.64 | 0.8451 | 0.8148 | 1.1829 | 1.4493 |
| *Tempete* | 24 | 1.65 | 38.68 | 32.37 | 0.9855 | 0.9565 | 0.8586 | 1.2874 |
| | 28 | 1.47 | 34.83 | 30.91 | 0.9716 | 0.9438 | 1.2056 | 1.5568 |
| | 32 | 1.18 | 30.88 | 29.19 | 0.9448 | 0.9276 | 1.6372 | 1.8330 |
| *Mobile* | 24 | 1.07 | 38.44 | 31.16 | 0.9871 | 0.9561 | 0.8652 | 1.4452 |
| | 28 | 0.91 | 34.51 | 29.83 | 0.9741 | 0.9443 | 1.2552 | 1.7373 |
| | 32 | 0.81 | 30.63 | 28.26 | 0.9501 | 0.9283 | 1.7154 | 2.0450 |
| *Hall* | 24 | 0.57 | 40.33 | 38.88 | 0.9744 | 0.9731 | 0.6536 | 0.7114 |
| | 28 | 0.48 | 37.92 | 37.19 | 0.9658 | 0.9649 | 0.7848 | 0.8276 |
| | 32 | 0.38 | 34.98 | 34.57 | 0.9527 | 0.9518 | 0.9542 | 0.9917 |
| *News* | 24 | 0.77 | 40.82 | 38.04 | 0.9848 | 0.9806 | 0.5847 | 0.6891 |
| | 28 | 0.62 | 37.78 | 36.20 | 0.9727 | 0.9693 | 0.7745 | 0.8552 |
| | 32 | 0.48 | 34.57 | 33.65 | 0.9531 | 0.9493 | 1.0064 | 1.0738 |

TABLE VI
TEST RESULTS BASED ON THE COMBINATION OF CAVLC CODEWORD MAPPING AND EXP-GOLOMB CODEWORD MAPPING

| *Sequence* | *QP* | *Maximum Capacity* (kbits/s) | *PSNR* (dB) | | *SSIM* | | *VQM* | |
|---|---|---|---|---|---|---|---|---|
| | | | *Original* | *Stegoed* | *Original* | *Stegoed* | *Original* | *Stegoed* |
| *Hall* | 24 | 1.17 | 40.33 | 38.97 | 0.9744 | 0.9731 | 0.6536 | 0.7093 |
| | 28 | 0.61 | 37.92 | 37.09 | 0.9658 | 0.9647 | 0.7848 | 0.8285 |
| | 32 | 0.40 | 34.98 | 34.62 | 0.9527 | 0.9519 | 0.9542 | 0.9844 |
| *News* | 24 | 1.27 | 40.82 | 37.85 | 0.9848 | 0.9802 | 0.5847 | 0.6906 |
| | 28 | 0.73 | 37.78 | 36.12 | 0.9727 | 0.9690 | 0.7745 | 0.8646 |
| | 32 | 0.50 | 34.57 | 33.84 | 0.9531 | 0.9503 | 1.0064 | 1.0630 |

codeword substituting method. For this type of video, it is appropriate to embed data using the codewords of *Levels*. According to Table IV, for low motion sequence (such as *Hall*, *News*), the embedding capacity is low if only the codewords of *Levels* are used for data hiding. In this case, both the *CAVLC* codewords of *Levels* and the Exp-Golomb codewords of *MVD*s can be used for data hiding. As depicted in Table V, for this type of video, the degradation in video quality caused by data hiding is quite small. So the combination is entirely feasible. The test results based on the combination of the *CAVLC* codewords of *Levels* and the Exp-Golomb codewords of *MVD*s are also given in Table VI. Compared with Table IV, the embedding capacity is improved obviously, but the video quality degradation is also negligible. Based on the above analysis, we can make a flexible choice of embedding carrier according to the situation.

To the best of our knowledge, till now, there is no algorithm to embed additional data directly in encrypted H.264/AVC video stream. Therefore, no detailed experimental comparisons are given in the paper. As described in Section I, in the existing related technologies [10]–[11], encryption and data hiding are accomplished almost simultaneously during H.264/AVC encoding process. In addition, encryption and data embedding would lead to increasing the bit-rate of H.264/AVC bitstream. On the contrary, our proposed scheme can encrypt H.264/AVC video stream directly and then embeds data into encrypted H.264/AVC video stream to meet the privacy-preserving requirements. The bit-rate of the encrypted H.264/AVC video stream containing hidden data is exactly the same as the original H.264/AVC video stream.

## V. 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 directly performed 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.

[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.

[7] X. P. Zhang, "Separable reversible data hiding in encrypted image," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 2, pp. 826–832, Apr. 2012.

[8] K. D. Ma, W. M. Zhang, X. F. Zhao, N. Yu, and F. Li, "Reversible data hiding in encrypted images by reserving room before encryption," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 3, pp. 553–562, Mar. 2013.

[9] A. V. Subramanyam, S. Emmanuel, and M. S. Kankanhalli, "Robust watermarking of compressed and encrypted JPEG2000 images," *IEEE Trans. Multimedia*, vol. 14, no. 3, pp. 703–716, Jun. 2012.

[10] S. G. Lian, Z. X. Liu, and Z. Ren, "Commutative encryption and watermarking in video compression," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 6, pp. 774–778, Jun. 2007.

[11] S. W. Park and S. U. Shin, "Combined scheme of encryption and watermarking in H.264/scalable video coding (SVC)," *New Directions Intell. Interact. Multimedia*, vol. 142, no. 1, pp. 351–361, 2008.

[12] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[13] S. G. Lian, Z. X. Liu, Z. Ren, and H. L. Wang, "Secure advanced video coding based on selective encryption algorithms," *IEEE Trans. Consumer Electron.*, vol. 52, no. 2, pp. 621–629, May 2006.

[14] Z. Shahid, M. Chaumont, and W. Puech, "Fast protection of H.264/AVC by selective encryption of CAVLC and CABAC for I and P frames," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 5, pp. 565–576, May 2011.

[15] M. N. Asghar and M. Ghanbari, "An efficient security system for CABAC bin-strings of H.264/SVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 3, pp. 425–437, Mar. 2013.

[16] T. Stutz and A. Uhl, "A survey of H.264 AVC/SVC encryption," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 3, pp. 325–339, Mar. 2012.

[17] *Advanced Video Coding for Generic Audiovisual Services*, ITU, Geneva, Switzerland, Mar. 2005.

[18] J. G. Jiang, Y. Liu, Z. P. Su, G. Zhang, and S. Xing, "An improved selective encryption for H.264 video based on intra prediction mode scrambling," *J. Multimedia*, vol. 5, no. 5, pp. 464–472, 2010.

[19] I. E. G. Richardson, *H.264 and MPEG-4 Video Compression: Video Coding for Next Generation Multimedia*. Hoboken, NJ, USA: Wiley, 2003.

[20] D. K. Zou and J. A. Bloom, "H.264 stream replacement watermarking with CABAC encoding," in *Proc. IEEE ICME*, Singapore, Jul. 2010, pp. 117–121.

[21] D. W. Xu and R. D. Wang, "Watermarking in H.264/AVC compressed domain using Exp-Golomb code words mapping," *Opt. Eng.*, vol. 50, no. 9, p. 097402, 2011.

[22] D. W. Xu, R. D. Wang, and J. C. Wang, "Prediction mode modulated data-hiding algorithm for H.264/AVC," *J. Real-Time Image Process.*, vol. 7, no. 4, pp. 205–214, 2012.

[23] T. Shanableh, "Data hiding in MPEG video files using multivariate regression and flexible macroblock ordering," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 2, pp. 455–464, Apr. 2012.


**Dawen Xu** received the M.S. degree in communication and information system from Ningbo University, Ningbo, China, and the Ph.D. degree in computer applied technology from Tongji University, Shanghai, China, in 2005 and 2011, respectively. He is an Associate Professor with the School of Electronics and Information Engineering, Ningbo University of Technology, Ningbo. His current research interests include digital watermarking and information hiding, and signal processing in the encrypted domain. He has been a Technical Paper Reviewer for IEEE conferences, journals, and magazines.

**Rangding Wang** received the M.S. degree from Northwestern Polytechnical University, Xi'an, China, in 1987, and the Ph.D. degree from Tongji University, Shanghai, China, in 2004. He is a Professor with the Faculty of Information Science and Engineering, Ningbo University, Ningbo, China. His current research interests include multimedia information security, digital speech processing, digital watermarking, steganography, and steganalysis. He is the author or coauthor of more than 120 research papers and two books. He holds more than 20 patents.

**Yun Q. Shi** (M'90–SM'93–F'05) received the B.S. and M.S. degrees from Shanghai Jiao Tong University, Shanghai, China, and the M.S. and Ph.D. degrees from the University of Pittsburgh, Pittsburgh, PA, USA. He has been a Professor with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, USA, since 1987. His current research interests include digital data hiding, steganalysis, forensics and information assurance, and visual signal processing and communications. He is the author or coauthor of more than 300 research papers, one book, and five book chapters, and an Editor of more than ten books. He holds 28 U.S. patents.

Dr. Shi received the Innovators Award 2010 from New Jersey Inventors Hall of Fame for Innovations in Digital Forensics and Security. His U.S. patent 7 457 341 titled "System and Method for Robust Reversible Data Hiding and Data Recovery in the Spatial Domain" won the 2010 Thomas Alva Edison Patent Award from the Research and Development Council of New Jersey. He served as an Associate Editor for two IEEE TRANSACTIONS and a few other journals.