# FAST PROTECTION OF H.264/AVC BY SELECTIVE ENCRYPTION OF CAVLC AND CABAC FOR I & P FRAMES

Z. Shahid, M. Chaumont and W. Puech

***Abstract*—This paper presents a novel method for the protection of bitstreams of state of the art video codec H.264/AVC. The problem of selective encryption (SE) is addressed along with the compression in the entropy coding modules. H.264/AVC supports two types of entropy coding modules. Context-adaptive variable length coding (CAVLC) is supported in H.264/AVC baseline profile and context-adaptive binary arithmetic coding (CABAC) is supported in H.264/AVC main profile. SE is performed in both types of entropy coding modules of this video codec. For this purpose, in this paper the encryption step is done simultaneously with the entropy coding CAVLC or CABAC. SE is performed by using the Advanced Encryption Standard (AES) algorithm with the Cipher Feedback (CFB) mode on a subset of *codewords/binstrings*. For CAVLC, SE is performed on equal length *codewords* from a specific variable length coding (VLC) table. In case of CABAC, it is done on equal length *binstrings*. In our scheme, entropy coding module serves the purpose of encryption cipher without affecting the coding efficiency of H.264/AVC by keeping exactly the same bitrate, generating completely compliant bitstream and utilizing negligible computational power. Owing to no escalation in bitrate, our encryption algorithm is better suited for real-time multimedia streaming over heterogeneous networks. It is perfect for playback on hand-held devices because of negligible increase in processing power. Nine different benchmark video sequences containing different combinations of motion, texture and objects are used for experimental evaluation of the proposed algorithm.**

***Index Terms*—Selective Encryption, CABAC, CAVLC, Video Security, AES Algorithm, Stream Cipher**

## I. INTRODUCTION

With the rapid growth of processing power and network bandwidth, many multimedia applications have emerged in the recent past. As digital data can easily be copied and modified, the concern about its protection and authentication have surfaced. Digital rights management (DRM) has emerged as an important research field to protect the copyrighted multimedia data. DRM systems enforce the rights of the multimedia property owners while ensuring the efficient rightful usage of such property.
Multimedia data requires either full encryption or selective encryption (SE) depending on the application requirements. For example military and law enforcement applications require full encryption. Nevertheless, there is a large spectrum of applications that demands security on a lower level, as for example that ensured by SE. SE encrypts part of the plaintext and has two main advantages. First, it reduces the computational requirements, since only a part of plaintext is encrypted [6]. Second, encrypted bitstream maintains the essential properties of the original bitstream [3]. SE just prevents abuse of the data. In the context of video, it refers to destroying the commercial value of video to a degree which prevents a pleasant viewing experience.

SE schemes based on H.264/AVC have been already presented on CAVLC [29] and CABAC [30]. These two previous methods fulfill real-time constraints by keeping the same bitrate and by generating completely compliant bitstream. In this paper, we have enhanced the previous proposed approaches by encryption of more syntax elements for CAVLC and extending it for P frames. Here we have also used AES [7] in the Cipher Feedback (CFB) mode which is a stream cipher algorithm. Security of the proposed schemes has also been analyzed in detail.
The rest of the paper is organized as follows. In Section II, overview of H.264/AVC and AES algorithm is presented. We explain the whole system architecture of the proposed methods in Section III. Section IV contains experimental evaluation and security analysis. In Section V, we present the concluding remarks about the proposed schemes.

Laboratory LIRMM, UMR CNRS 5506, University of Montpellier II, 161, rue Ada, 34392 MONTPELLIER Cedex 05, FRANCE zafar.shahid@lirmm.fr, marc.chaumont@lirmm.fr, william.puech@lirmm.fr

## II. Description of the H.264/AVC-based video encryption system

### A. Overview of H.264/AVC

H.264/AVC (also known as MPEG4 Part 10) [1] is state of the art video coding standard of ITU-T and ISO/IEC. In H.264/AVC, an input video frame is divided into macro-blocks (MB) of 16x16 pixels and each of them is encoded separately. Each video frame can be encoded as *intra* (I frame) or *inter* (P and B frames). In I frame, the current MB is predicted spatially from MBs which have been previously encoded and reconstructed (MB at top and left). In P frame, motion compensated prediction is done from the previous reference frames, while bidirectional prediction from both previous and next reference frames is performed for B frames. The purpose of the reconstruction in the encoder is to ensure that both the encoder and decoder use identical reference frames to create the predictions. If this is not the case, then the predictions in encoder and decoder will not be identical, leading to an increasing error or ”drift” between the encoder and decoder. The difference between original and predicted frame is call residual. This residual is coded using transform coding followed by quantization and zigzag scan. In the last step, entropy coding comes into action. Quantized transform coefficients are then coded using either CAVLC [4] or CABAC [20]. The block diagram of H.264/AVC is shown in Fig. 1. On the decoding side, compressed bitstream is decoded by entropy decoding module, followed by inverse-zigzag scan. These coefficients are then rescaled and inverse transformed to get the residual signal which is added to the predicted signal to get the decoded video frame. H.264/AVC has some additional features as compared to previous video standards including MPEG2 and MPEG4 Part II. In *baseline* profile of H.264/AVC, it has $4 \times 4$ integer transform (IT) in contrast to $8 \times 8$ transform of previous standards. In higher profiles, it offers transform coding for adaptive size. DCT transform has been replaced by IT which does not need any multiplication operation and can be implemented by only additions and shifts and thus requires lesser number of computations. For *Inter* frame, H.264/AVC supports variable block size motion estimation, quarter pixel accuracy, multiple reference frames, improved skipped and direct motion inference. For *Intra* frame, it offers additional spatial prediction modes. All these additional features of H.264/AVC are aimed to outperform previous video coding standards [35].

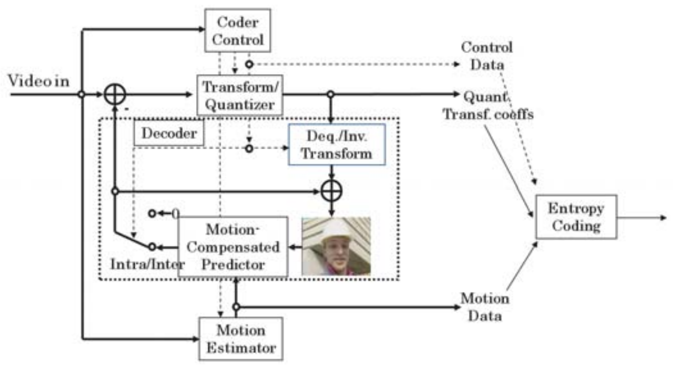


Fig. 1: Block diagram of H.264/AVC video encoder.

H.264 scans the non-zero coefficients (NZs) in inverse zigzag order (from high frequency NZs to low frequency NZs) which are then passed to entropy coding module. We review the basic working of CAVLC in Section II-A1 and of CABAC in Section II-A2.

*1) Context-Adaptive Variable Length Coding:* In CAVLC, Run-length coding is performed first as it encodes levels and runs separately. CAVLC is designed to exploit the characteristics of NZs and works in several steps.

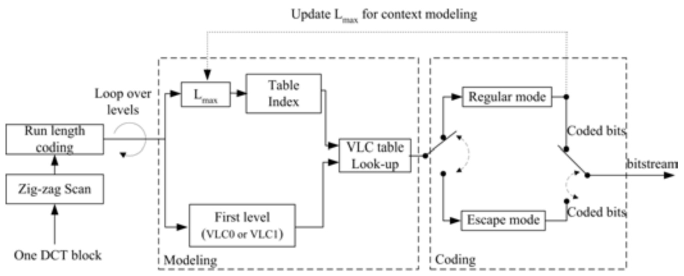


Fig. 2: Block diagram of level coding in CAVLC of H.264/AVC.

To adapt to the local statistical features of DCT coefficients, CAVLC uses seven fixed variable length coding (VLC) tables. For example, $'2'$ will be coded as $'010'$ using VLC1 table, while it will be coded as $'1010'$ using VLC3 table. If magnitude of NZ lies within the range of that VLC table, it is coded by regular mode, otherwise escape mode is used. Adaptive nature is introduced by changing the table for the next NZ based on the magnitude of the current NZ as shown in Fig. 2. For the first NZ, VLC0 table is used unless there are more than 10 NZs and less than 3 trailing ones, in which case it is coded with VLC1 table. The tree representation of first four VLC tables is shown in Fig. 3.

*2) Context-Adaptive Binary Arithmetic Coding:* CABAC is designed to better exploit the characteristics of NZs as compared to CAVLC, consumes more processing and offers about 10% better compression than CAVLC on average [22]. Run-length coding has been replaced by *significant map* coding which specifies the

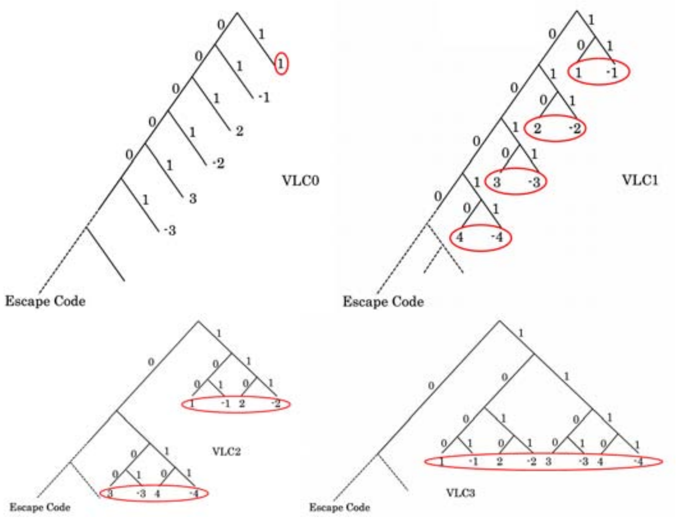
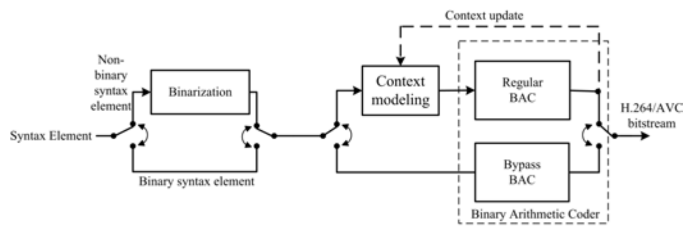
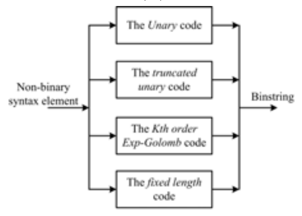


Fig. 3: Tree representation of first four VLC tables used in CAVLC. In each VLC table, VLC codes for encircled coefficients have same code length.



(a)



(b)

Fig. 4: (a) Block diagram of CABAC of H.264/AVC, (b) Binarization stage.

position of NZs in the 4x4 block. Binary arithmetic coding module (BAC) of CABAC uses many context models to encode NZs and context model for a specific NZ depends on recently coded NZs.

CABAC consists of multiple stages as shown in Fig. 4.a. First of all, *binarization* is done in which, non-binary syntax elements are converted to binary form called *binstrings* which are more amenable to compression by BAC. Binary representation for a non-binary syntax element is done in such a way that it is close to minimum redundancy code. In CABAC, there are four basic code trees for *binarization* step, namely the *unary* code, the *truncated unary* code, the *kth order Exp-Golomb* code (EGk) and the *fixed length* code as shown in Fig. 4.b.

For an unsigned integer value $x \geq 0$, the *unary* code consists of $x$ 1's plus a terminating 0 bit. The *truncated unary* code is only defined for $x$ with $0 \leq x \leq s$. For $x < s$ the code is given by the *unary* code, whereas for $x = s$ the terminating "0" bit is neglected. EGk is constructed by a concatenation of a prefix and a suffix parts and is suitable for binarization of syntax elements that represent prediction residuals. For a given unsigned integer value $x > 0$, the prefix part of the EGk *binstring* consists of a unary code corresponding to the length $l(x) = \left[log_2(\frac{x}{2k}+1)\right]$. The EGk suffix part is computed as the binary representation of $x+2^k(1-2^{l(x)})$ using $k + l(x)$ significant bits. Consequently for EGk binarization, the code length is $2l(x)+k+1$. When k = 0, $2l(x)+k+1 = 2l(x)+1$.

The *fixed length* code is applied to syntax elements with a nearly uniform distribution or to syntax elements, for which each bit in the *fixed length* code *binstring* represents a specific coding decision e.g., *coded block flag*. Three syntax elements are binarized by concatenation of the basic code trees, namely *coded block pattern*, NZ and the motion vector difference (MVD). Binarization of absolute level of NZs is done by concatenation of *truncated unary* code and EG0. The *truncated unary* code constitutes the prefix part with cutoff value $S = 14$. Binarization and subsequent arithmetic coding process is applied to the syntax element $coeff\_abs\_value\_minus1 = abs\_level - 1$, since quantized transformed coefficients with zero magnitude are encoded using *significant map*. For MVD, *binstring* is constructed by concatenation of the *truncated unary* code and EG3. The *truncated unary* constitutes the prefix part with cutoff value $S = 9$. Suffix part of MVDs contains EG3 of $|MVD| - 9$ for $|MVD| > 9$ and sign bit.

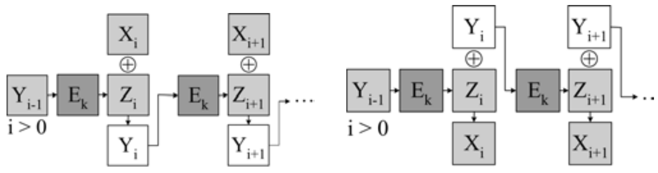
## B. The AES encryption algorithm

The Advanced Encryption Standard (AES) algorithm consists of a set of processing steps repeated for a number of iterations called rounds [7]. The number of rounds depends on the size of the key and the size of the data block. The number of rounds is 9 for example, if both the block and the key are 128 bits long. Given a sequence $\{X_1, X_2, ..., X_n\}$ of bit plaintext blocks, each $X_i$ is encrypted with the same secret key $k$ producing the ciphertext blocks $\{Y_1, Y_2, ..., Y_n\}$. To encipher a data block $X_i$ in AES you first perform an AddRoundKey step by XORing a subkey with the block. The incoming data and the key are added together in the first AddRoundKey step. Afterward, it follows the round operation. Each regular round operation involves four steps which are SubBytes, ShiftRows, MixColumns and AddRoundKey. Before producing the final ciphered data $Y_i$, the AES performs an extra final routine that is composed of SubBytes, ShiftRows and AddRoundKey steps.

The AES algorithm can support several cipher modes: ECB (Electronic Code Book), CBC (Cipher Block Chaining), OFB (Output Feedback), CFB (Cipher Feedback) and CTR (Counter) [31]. The ECB mode is actually the basic AES algorithm. With the ECB mode, each plaintext block $X_i$ is encrypted with the same secret key $k$ producing the ciphertext block $Y_i = E_k(X_i)$. The CBC mode adds a feedback mechanism to a block cipher. Each ciphertext block $Y_i$ is XORed with the incoming plaintext block $X_{i+1}$ before being encrypted with the key $k$. An initialization vector (IV) is used for the first iteration. In fact, all modes (except the ECB mode) require the use of an IV. In CFB mode, as shown in Fig. 5, the keystream element $Z_i$ is generated and the ciphertext block $Y_i$ is produced as:

$$\begin{cases} Z_i &= E_k(Y_{i-1}), for\ i \geq 1 \\ Y_i &= X_i \oplus Z_i \end{cases}, \quad (1)$$

where $\oplus$ is the XOR operator.



(a) (b)

Fig. 5: CFB stream cipher: (a) Encryption, (b) Decryption.

In the OFB mode, $Z_0$ is substituted by the IV and the input data is encrypted by XORing it with the output $Z_i$. The CTR mode has very similar characteristics to OFB, but in addition it allows pseudo random access for decryption. It generates the next keystream block by encrypting successive values of a counter.

Although AES is a block cipher, in the OFB, CFB and CTR modes it operates as a stream cipher. These modes do not require any special measures to handle messages whose lengths are not multiples of the block size since they all work by XORing the plaintext with the output of the block cipher. Each mode has its advantages and disadvantages. For example in ECB and OFB modes, any modification in the plaintext block $X_i$ causes the corresponding ciphered block $Y_i$ to be altered, but other ciphered blocks are not affected. On the other hand, if a plaintext block $X_i$ is changed in CBC and CFB modes, then $Y_i$ and all subsequent ciphered blocks will be affected. These properties mean that CBC and CFB modes are useful for the purpose of authentication while ECB and OFB modes treat separately each block. Therefore, we can notice that OFB does not spread noise, while the CFB does that.

## C. SE of image and video

Selective encryption (SE) is a technique aiming to save computational time or to enable new system functionalities by only encrypting a portion of a compressed bitstream while still achieving adequate security [18]. SE as well as partial encryption (PE) are applied only on certain parts of the bitstream. In the decoding stage, both the encrypted and the non-encrypted information should be appropriately identified and displayed [6], [21], [26]. The copyright protection of the multimedia content is a required feature for DRM systems. The technical challenges posed by such systems are high and previous approaches have not entirely succeeded in tackling them [17].

In [32], Tang proposed a technique called zigzag permutation applicable to DCT-based image and video codecs. On one hand this method provides a certain level of confidentiality, while on the other hand it increases the overall bitrate. For image, several SE techniques have bee proposed in literature. In [8], Droogenbroeck and Benedett proposed a technique for encryption of JPEG images. It encrypts a selected number of AC coefficients. The DC coefficients are not ciphered since they carry important visual information and they are highly predictable. In spite of the constancy in the bitrate while preserving the bitstream compliance, the compression and the encryption process are separated and consequently the computational complexity is increased.

The Advanced Encryption Standard (AES) [7] has been used for SE of image and video in literature. The AES was applied on the Haar discrete wavelet transform compressed images in [23]. The encryption of color images in the wavelet transform has been addressed in [21]. In this approach the encryption is performed on the resulting wavelet code bits. In [25], SE was performed on color JPEG images by selectively encrypting only *luma* component using AES cipher. The protection rights of individuals and the privacy of certain moving objects in the context of security surveillance systems using viewer generated masking and the AES encryption standard has been addressed in [37].

Combining PE and image/video compression using the set partitioning in hierarchical trees was used in [6].

Nevertheless, this approach requires a significant computational complexity. A method that does not require significant processing time and which operates directly on the bit planes of the image was proposed in [19]. The robustness of partially encrypted videos to attacks which exploit the information from non-encrypted bits together with the availability of side information was studied in [27]. Fisch *et al.* [10] proposed a scalable encryption method for a DCT-coded visual data wherein the data are organized in a scalable bitstream form. These bitstreams are constructed with the DC and some AC coefficients of each block which are then arranged in layers according to their visual importance and PE process is applied over these layers.

For video, there are several SE techniques for different video codecs presented in literature. SE of MPEG4 video standard was studied in [34] wherein Data Encryption Standard (DES) was used to encrypt fixed length and variable length codes. In this approach, the encrypted bitstream is completely compliant with MPEG4 bitstream format but it increases the bitrate. A trade off has to be made among complexity, security and the bit overhead. In [38], SE of MPEG4 video standard is proposed by doing frequency domain selective scrambling, DCT block shuffling and rotation. This scheme is very easy to perform but its limitation is its bitrate overhead. SE of ROI of MPEG4 video has been presented in [9]. It performs SE by pseudo randomly inverting sign of DCT coefficients in ROI. SE of H.264/AVC has been studied in [15] wherein encryption has been carried out in some fields like intra-prediction mode, residual data, inter-prediction mode and motion vectors. A scheme for commutative encryption and watermarking of H.264/AVC is presented in [16]. Here SE of some MB header fields is combined with watermarking of magnitude of DCT coefficients. This scheme presents a watermarking solution in encrypted domain without exposing video content. The limitation of techniques proposed in [15], [16] is that they are not format compliant. Encryption for H.264/AVC has been discussed in [5] wherein they do permutations of the pixels of MBs which are in *region of interest* (ROI). The drawback of this scheme is that bitrate increases as the size of ROI increases. This is due to change in the statistics of ROI as it is no more a slow varying region which is the basic assumption for video signals.
SE of H.264/AVC at network abstraction layer (NAL) has been proposed by [14]. Important NAL units namely instantaneous decoding refresh (IDR) picture, sequence parameter set (SPS), and picture parameter set (PPS) are encrypted with a stream cipher. The limitation of this scheme is that it is not format compliant and cannot be parsed even at frame level. SE of H.264/AVC using AES has been proposed in [2]. In this scheme, encryption of I frame is performed, since P and B frame are not significant without I frames.. This scheme is not format compliant.
The use of general entropy coder as encryption cipher using statistical models has been studied in the literature in [36]. It encrypts by using different Huffman tables for different input symbols. The tables, as well as the order in which they are used, are kept secret. This technique is vulnerable to known plaintext attacks as explained in [12]. Key-based interval splitting of arithmetic coding (KSAC) has used an approach [13] wherein intervals are partitioned in each iteration of arithmetic coding. Secret key is used to decide how the interval will be partitioned. Number of sub intervals in which an interval is divided should be kept small as it increases the bitrate of bitstream. Randomized arithmetic coding [11] is aimed at arithmetic coding but instead of partitioning of intervals like in KSAC, secret key is used to scramble the order of intervals. The limitation of these entropy coding based techniques is that encrypted bitstream is not format compliant. Moreover, these techniques require lot of processing power.

In the context of DRM systems, our study addresses the simultaneous SE and compression for state of the art H.264/AVC. The encrypted bitstream is format compliant with absolutely no escalation in bitrate. Furthermore, it does not require lot of processing power for encryption and decryption. In Section III we describe our proposed approaches to apply simultaneously SE and H.264/AVC compression in video sequences.

## III. The proposed selective encryption schemes

Our approach consists of SE during the entropy coding stage of H.264/AVC as shown in Fig. 6. In baseline profile, SE is performed in CAVLC entropy coding stage (SE-CAVLC). While in main profile, it is performed in CABAC entropy coding stage (SE-CABAC). In SE of video, encrypted bitstream compliance is a required feature for some direct operations such displaying, time seeking and browsing. Encrypted bitstream will be compliant and fulfills real-time constraints if the following three conditions are fulfilled:

- To keep the bitrate of encrypted bitstream same as the original bitstream, encrypted *codewords/binstrings* must have the same size as the original *codewords/binstrings*.
- The encrypted *codewords/binstrings* must be valid so that they may be decoded by entropy decoder.

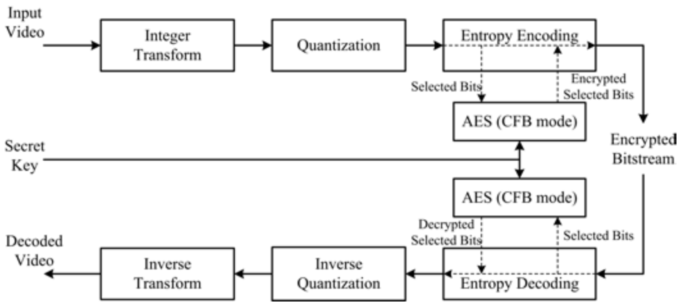


Fig. 6: Block diagram of encryption and decryption process in H.264/AVC.

- The decoded value of syntax element from encrypted *codewords/binstrings* must stay in the valid range for that syntax element. Any syntax element which is used for prediction of neighboring MBs should not be encrypted. Otherwise the drift in the value of syntax element will keep on increasing and after a few iterations, value of syntax element will fall outside the valid range and bitstream will be no more decodable.

In each MB, header information is encoded first, which is followed by the encoding of MB data. To keep the bitstream compliant, we cannot encrypt MB header, since it is used for prediction of future MBs. MB data contains NZs and can be encrypted. A MB is further divided into 16 blocks of 4x4 pixels to be processed by IT module. The *coded block pattern* is a syntax element used to indicate which 8x8 blocks within a MB contain NZs. The *macroblock mode* (MBmode) is used to indicate whether a MB is *skipped* or not. If MB is not *skipped*, then MBmode indicates the prediction method for a specific MB. For a 4x4 block inside MB, if *coded block pattern* and MBmode are set, it indicates that this block is encoded. Inside 4x4 block, *coded block flag* is the syntax element used to indicate whether it contains NZs or not. It is encoded first. If it is zero, no further data is transmitted; otherwise, it is followed by encoding of *significant map* in case of CABAC. Finally, the absolute value of each NZ and its sign are encoded. Similar to MB header, header of 4x4 block which includes *coded block flag* and *significant map*, should not be encrypted for the sake of bitstream compliance.
Available encryption space (ES) which fulfills the above mentioned conditions for SE-CAVLC and SE-CABAC is presented in Section III-A and III-B respectively. Encryption and decryption of the protected bitstream are presented in Section III-C and III-D respectively.

## A. Encryption space (ES) for SE-CAVLC

In CAVLC, five syntax elements are used to code levels and runs as shown in Fig. 7. NZs are coded by three syntax elements namely coeff_token, signs of trailing ones and remaining non-zero levels. Zeros are coded by two syntax elements namely total no. of zeros and runs of zeros. A single syntax element namely coeff_token is used to code total NZs and number of trailing ones. It is followed by coding of signs of trailing ones (T1's). Remaining NZs are then coded using seven VLC look-up tables either by regular mode or by escape mode as explained in Section II-A1. They are mapped to some code from a specific VLC look-up table.

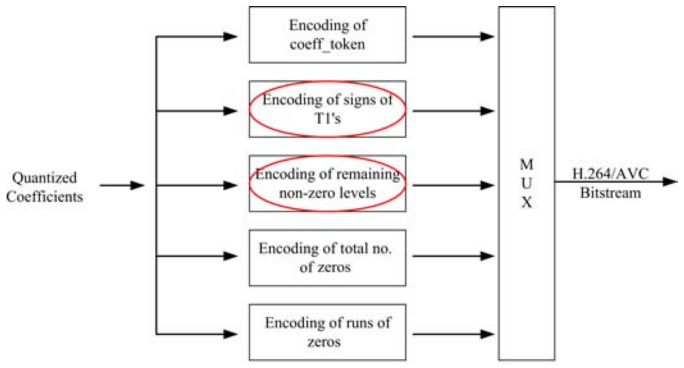

Quantized Coefficients
Encoding of coeff_token
Encoding of signs of T1's
Encoding of remaining non-zero levels
Encoding of total no. of zeros
Encoding of runs of zeros
M U X
H.264/AVC Bitstream

Fig. 7: Block diagram of CAVLC of H.264/AVC. Encircled syntax elements are used for SE-CAVLC.

To keep the bitstream compliant, we cannot encrypt coeff_token, total number of zeros and runs of zeros. Two syntax elements fulfill the above mentioned conditions for encryptions. First is signs of trailing ones. Second is sign and magnitude of remaining NZs, both in regular and escape mode. For the sake of same bitrate, ES of SE-CAVLC consists of only those NZs whose VLC *codewords* have the same length. CAVLC uses multiple VLC tables with some threshold for incrementing the table as given in equation (2). Since the threshold for a specific table is highest possible value possible with that *codeword* length (this is the case when all the suffix bits of the *codeword* are 1), magnitude of encrypted NZ is such that VLC table transition is not affected. VLC codes, having same code length, constitute the ES. For VLC$n$ table, ES is $2^n$ as given in equation (3). For table VLC0, every NZ has different *codeword* length, consequently we cannot encrypt the NZs in this table:

$$TH[0\dots6] = (0,\ 2,\ 3,\ 6,\ 12,\ 24,\ 48,\ \infty). \quad (2)$$

$$ES[0\dots6] = (1,\ 2,\ 4,\ 8,\ 16,\ 32,\ 64,\ \infty). \quad (3)$$

## B. Encryption space (ES) for SE-CABAC

The main difference between SE-CAVLC and SE-CABAC is that in SE-CABAC, SE is not performed on CABAC bitstream. Rather it is performed on *binstrings* which are input to BAC as shown in Fig. 8. Among all the four *binarization* techniques, the *unary* and *truncated unary* codes have different code lengths for each input value as explained in Section II-A2. They do not fulfill the first condition and their encryption will change the bitrate of bitstream. Suffix of EGk and the *fixed length* code can be encrypted while keeping the bitrate unchanged. EGk is used for binarization of absolute value of levels and MVDs. Number of MVD *binstrings* have the same length and hence, first and second conditions are fulfilled. But owing to the fact that MVDs are part of MB header and are used for prediction of future motion vectors, their encryption does not fulfill third condition and their encryption makes the bitstream non-compliant. To conclude, the syntax elements which fulfill the criteria for encryption of H.264/AVC compliant bitstream are suffix of EG0 and sign bits of levels. Hence for each NZ with $|NZ| > 14$, encryption is performed on $l(x)$ of EG0. It is followed by encryption of syntax element *coeff_sign_flag* which represents sign of levels of all non-zero levels. The *fixed length* code is used for binarization of syntax elements which belong to MB header and cannot be encrypted.
To keep the bitrate intact, ES for SE-CABAC consists of only those NZs whose EG0 *binstrings* have the same length as shown in Fig. 9. EG0 codes, having same code length, constitute the ES and it depends upon $\|NZ\|$. The ES is $2^{log_2(n+1)}$ where $n$ is the maximum possible value by suffix bits of EG0 *i.e.* when all the bits in suffix are 1.

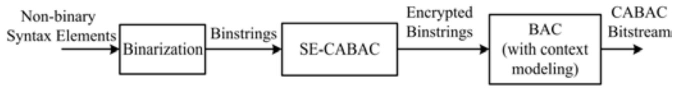


Fig. 8: SE of *binstrings* in SE-CABAC.

## C. SE of NZs in the entropy coding stage of H.264/AVC

Let us consider $Y_i = X_i \oplus E_k(Y_{i-1})$ as the notation for the encryption of a $n$ bit block $X_i$, using the secret key $k$ with the AES cipher in CFB mode as given by equation (1), and performed as described in the scheme from Fig. 5. We have chosen to use this mode in order to keep the original compression rate. Indeed, with the CFB mode for each block, the size of the encrypted data $Y_i$ can be exactly the same one as the size of the plaintext $X_i$. In

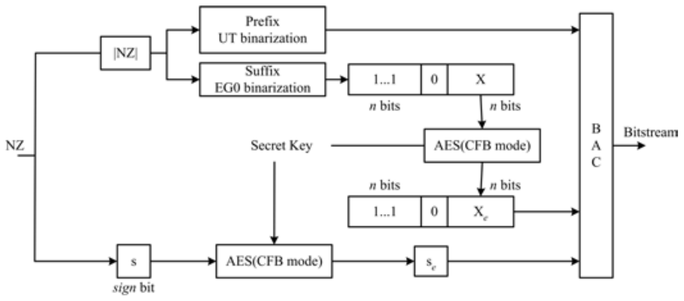


Fig. 9: Encryption process for NZs and their signs in CABAC of H.264/AVC.

this mode, the code from the previously encrypted block is used to encrypt the current one as shown in Fig. 5. The three stages of the proposed algorithm are: the construction of the plaintext $X_i$, described in Section III-C1, the encryption of $X_i$ to create $Y_i$ which is provided in Section III-C2 and the substitution of the original *code-word/binstring* with the encrypted information, which is explained in Section III-C3. The overview of the proposed SE method is provided in Fig. 10.

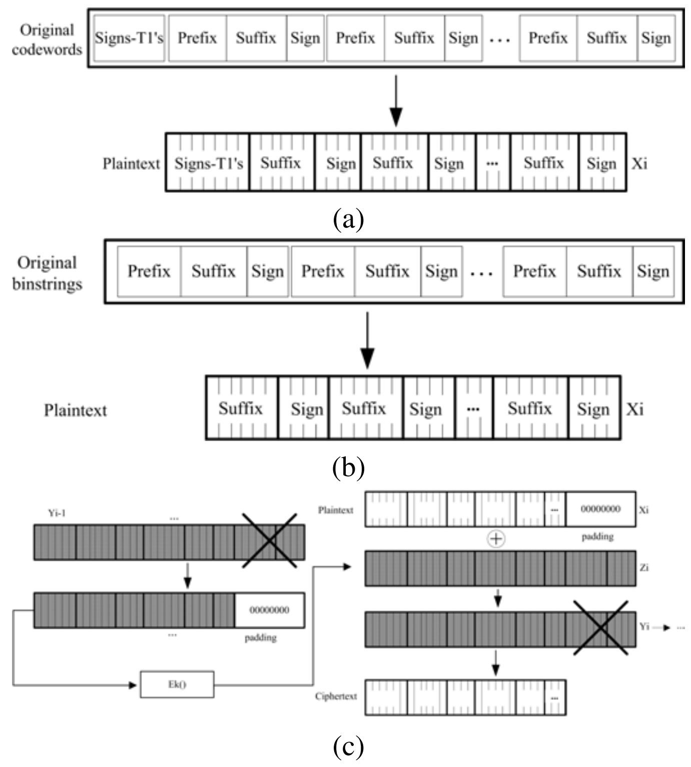


Fig. 10: Global overview of the proposed SE method (a) Preparation of plaintext for CAVLC, (b) Preparation of plaintext for CABAC, (c) Proposed SE scheme.

*1) The construction of plaintext:* As slices are independent coding units, SE should be performed on them independently. In case of SE-CAVLC, the plaintext is created by copying the encrypt-able bits from CAVLC bitstream to the vector $X_i$ until either $X_i$ is completely filled or slice-boundary comes as shown in Fig. 10.a. Let $C$, the length of the vector $X_i$, is 128. In case of SE-CABAC, we perform SE before BAC as shown in Fig. 10.b. In that case, we transform the non-binary syntax elements to *binstrings* through process of binarization and at the same time we fill the $X_i$ with encrypted bits until either the vector $X_i$ is completely filled or the slice boundary comes. The binarization of many syntax elements at the same time also makes the CABAC coding faster and increases its throughput [39].

Let $L(X_i)$ be the length up to which vector $X_i$ is filled. In case of slice boundary, if $L(X_i) < C$, we apply a padding function $p(j) = 0$, where $j \in \{L(X_i) + 1, \dots, C\}$, to fill in the vector $X_i$ with zeros up to $C$ bits. Historically, padding was used to increase the security of the encryption, but in here it is used for rather technical reasons [28].

*2) Encryption of the plaintext with AES in the CFB mode:* In the encryption step with AES in the CFB mode, the previous encrypted block $Y_{i-1}$ is used as the input of the AES algorithm in order to create $Z_i$. Then, the current plaintext $X_i$ is XORed with $Z_i$ in order to generate the encrypted text $Y_i$ as given by equation (1). For the initialization, the IV is created from the secret key $k$ according to the following strategy. The secret key $k$ is used as the seed of the pseudo random number generator (PRNG). Firstly, the secret key $k$ is divided into 8 bits (byte) sequences. The PRNG produces a random number for each byte component of the key that defines the order of IV formation. Then, we substitute $Y_0$ with the IV, and $Y_0$ is used in AES to produce $Z_1$.

As illustrated in Fig. 10.c, with the CFB mode of the AES algorithm, the generation of the keystream $Z_i$ depends on the previous encrypted block $Y_{i-1}$. Consequently, if two plaintexts are identical $X_i = X_j$ in the CFB mode, then always the two corresponding encrypted blocks are different, $Y_i \neq Y_j$.

*3) Substitution of the original bitstream:* The third step is the substitution of the original $Y_i$ by the encrypted $Y_i$. For SE-CAVLC, CAVLC bitstream is accessed in sequential order as in the first step (construction of the plaintext $X_i$). Given the length in bits of each amplitude $(S_n, S_{n-1}, \dots, S_1)$, we start substituting the original bits in the bitstream by the corresponding parts of $Y_i$ as shown in Fig. 10. For SE-CABAC, *binstrings* are accessed in sequential order and we start substituting the original bits in them by the corresponding parts of $Y_i$ as shown in Fig. 10. In case of slice boundaries, the total quantity of replaced bits is $L(X_i)$ and consequently we do not necessarily use all the bits of $Y_i$.

### *D. Decryption process*

The decryption process in the CFB mode works as follows. The previous block $Y_{i-1}$ is used as the input to the AES algorithm in order to generate $Z_i$. By knowing the secret key $k$, we apply the same function $E_k(\cdot)$ as that used in the encryption stage. The difference is that the input of this process is now the ciphered vector. In case of SE-CAVLC, the ciphered vector is accessed in the sequential way in order to construct the plaintext $Y_{i-1}$ which is then used in the AES to generate the keystream $Z_i$. The keystream $Z_i$ is then XORed with the current block $Y_i$ to generate $X_i$, as shown in Fig. 5.b. For SE-CAVLC, the resulting plaintext vector is split into segments in order to substitute the signs of trailing ones and suffixes $(S_n, S_{n-1}...S_1)$ in the ciphered bitstream and to generate the original CAVLC bitstream. Afterward, we apply the entropy decoding and retrieve the quantized DCT coefficients. After the inverse quantization and the inverse DCT we get the decrypted and decoded video frame.

In case of SE-CABAC, the difference is that binary arithmetic decoder is used to transform the SE-CABAC bitstream to encrypted *binstrings* which are then accessed to make the plaintext $Y_{i-1}$. The plaintext is decrypted and substituted back to generate original *binstrings*. They are then passed through inverse binarization, inverse quantization and inverse DCT steps to get the decrypted and decoded video frame.

## IV. Experimental results

In this section we analyze the results for SE-CAVLC and SE-CABAC. We have used the reference implementation of H.264 JSVM 10.2 in AVC mode for video sequences in QCIF and SD resolution. For the experimental results, nine benchmark video sequences have been used for the analysis in QCIF format. Each of them represents different combinations of motion (fast/slow, pan/zoom/rotation), color (bright/dull), contrast (high/low) and objects (vehicle, buildings, people). The video sequences 'bus', 'city' and 'foreman' contain camera motion while 'football' and 'soccer' contain camera panning and zooming along with object motion and texture in background. The video sequences 'harbour' and 'ice' contain high luminance images with smooth motion. 'Mobile' sequence contains a complex still background and foreground motion.

In Section IV-A we present an analysis of joint SE and H.264/AVC compression while in Section IV-B we compare PSNR and quality when applying SE only on I frames and on I+P frames. In Section IV-C, security analysis, showing the efficiency of the proposed method, is developed[1].

## *A. Analysis of joint SE and H.264/AVC compression*

We have applied simultaneously our SE and H.264/AVC compression as described in Section III, on all the benchmark video sequences. SE-CAVLC and SE-CABAC impart some characteristics to the bitstream. In spatial domain, SE video gets flat regions and change in pixel values mostly occur on MB boundaries. In temporal domain, *luma* and *chroma* values rise up to maximum limit and then come back to minimum values. This cycle keeps on repeating. Owing to this phenomenon, the pixel values change drastically in temporal domain. Lot of transitions are observed in values of color and brightness. This phenomenon can be observed for SE-CAVLC and SE-CABAC in Fig. 11 and Fig. 12 respectively for QP value 18 for *foreman* video sequence.

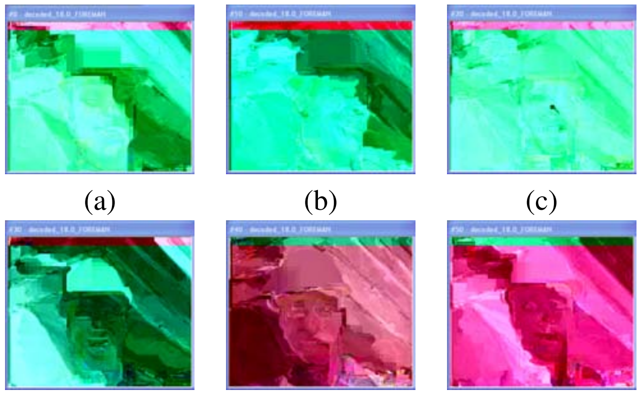

(a) (b) (c)
(d) (e) (f)

Fig. 11: Six frames of *foreman* video sequence for SE-CAVLC for QP value 18 with frame: a) #0, b) #10, c) #20, d) #30, e) #40, f) #50.

In a first set of experiments, we have analyzed the available encryption space (ES) in H.264/AVC bitstreams for both of SE-CAVLC and SE-CABAC. ES is defined as percentage of total bitstream size. MBs that contain many details and texture will have lot of NZs and consequently, will be strongly encrypted. On the other hand, the homogeneous MBs, *i.e.* blocks that contain series of identical pixels, are less ciphered because they contain a lot of null coefficients which are represented by runs in CAVLC and by significant map in CABAC. In Table I, we provide ES for SE-CAVLC and SE-CABAC for different benchmark video sequences for QP value 18. While in Table II, ES for various QP

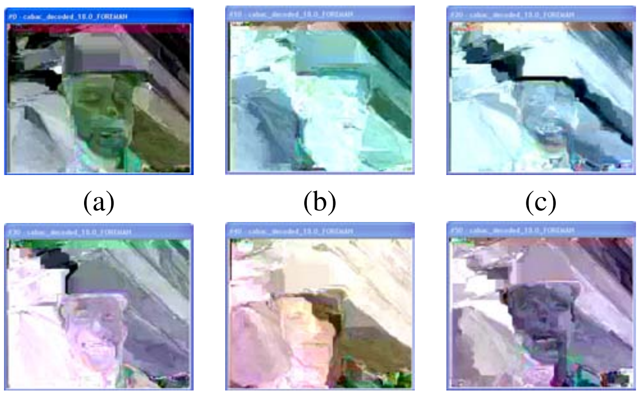

(a) (b) (c)
(d) (e) (f)

Fig. 12: Six frames of *foreman* video sequence for SE-CABAC for QP value 18 with frame: a) #0, b) #10, c) #20, d) #30, e) #40, f) #50.

values is shown for *foreman* video sequence. Here the average number of bits available for SE per MB are also provided. One can note that ES is inversely proportional to QP value. When QP value is higher and implicitly the video compression is higher, we are able to encrypt fewer bits in the compressed frame. This is due to the fact that H.264/AVC has lesser number of NZs at higher QP values. From both these tables, it is evident that more ES is available for SE-CAVLC as compared to SE-CABAC. But ES is more affected by change in QP values for SE-CAVLC as compared to SE-CABAC. For example, for *foreman* video sequence, ES varies from $28.55\%$ to $6.70\%$ for SE-CAVLC when QP varies from 12 to 42. For the same QP range, the change in ES for SE-CABAC is from $19.97\%$ to $9.46\%$ as shown in Table II. From Table I and II, since PSNR of original H.264/AVC are very similar for both CAVLC and CABAC, in the rest of this section for the sake of comparison, we list only PSNR of CAVLC bitstreams.

| | SE-CAVLC | | | | SE-CABAC | | | |
|---|---|---|---|---|---|---|---|---|
| | encoder | | decoder | | encoder | | decoder | |
| Seq. | I (%) | I+P (%) | I (%) | I+P (%) | I (%) | I+P (%) | I (%) | I+P (%) |
| bus | 0.69 | 0.31 | 3.77 | 2.7 | 0.57 | 0.25 | 3.37 | 2.3 |
| city | 0.5 | 0.26 | 3.36 | 2.4 | 0.44 | 0.23 | 3.06 | 2.1 |
| crew | 0.31 | 0.15 | 2.52 | 1.5 | 0.29 | 0.14 | 2.22 | 1.2 |
| football | 0.41 | 0.23 | 3.46 | 2.4 | 0.31 | 0.18 | 3.26 | 2.2 |
| foreman | 0.47 | 0.23 | 3.19 | 2.2 | 0.41 | 0.20 | 2.99 | 2.0 |
| harbour | 0.55 | 0.30 | 3.65 | 2.7 | 0.47 | 0.26 | 3.25 | 2.3 |
| ice | 0.41 | 0.21 | 3.16 | 2.1 | 0.33 | 0.17 | 2.96 | 1.9 |
| mobile | 0.76 | 0.35 | 4.33 | 3.3 | 0.72 | 0.33 | 4.03 | 3.0 |
| soccer | 0.44 | 0.21 | 3.17 | 2.2 | 0.38 | 0.18 | 2.87 | 1.9 |

TABLE III: Analysis of increase in processing power for SE-CAVLC and SE-CABAC at QP value 18.

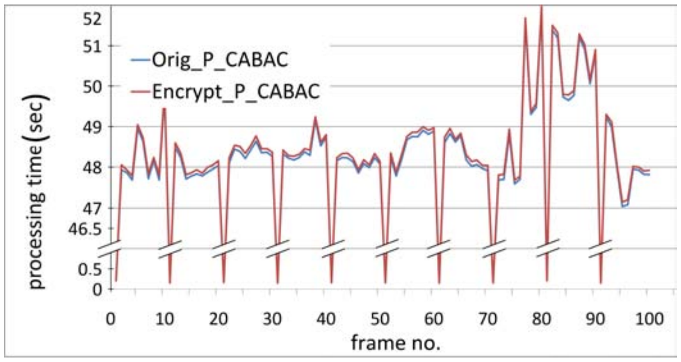
[1]Encrypted video bitstreams are available on http://www.lirmm.fr/~shahid/.

| Seq. | SE-CAVLC PSNR (dB) | SE-CAVLC Total Size (Bytes) | SE-CAVLC *ES* (%) | SE-CAVLC Avg. *ES* Bits/MB. | SE-CABAC PSNR (dB) | SE-CABAC Total Size Bytes | SE-CABAC *ES* (%) | SE-CABAC Avg. *ES* Bits/MB. |
|---|---|---|---|---|---|---|---|---|
| bus | 44.25 | 1254523 | 31.05 | 39 | 44.24 | 1255497 | 19.93 | 25 |
| city | 44.29 | 1022852 | 26.41 | 27 | 44.27 | 1024053 | 19.79 | 20 |
| crew | 44.82 | 779480 | 20.66 | 16 | 44.81 | 777037 | 18.97 | 15 |
| football | 44.61 | 997640 | 25.33 | 26 | 44.59 | 987936 | 19.45 | 19 |
| foreman | 44.38 | 813195 | 22.76 | 19 | 44.36 | 806063 | 18.72 | 15 |
| harbour | 44.10 | 1279309 | 30.49 | 39 | 44.09 | 1268153 | 20.01 | 26 |
| ice | 46.47 | 472573 | 24.64 | 12 | 46.46 | 469323 | 17.72 | 8 |
| mobile | 44.44 | 1768771 | 36.17 | 65 | 44.43 | 1753381 | 19.80 | 35 |
| soccer | 44.27 | 922527 | 23.42 | 22 | 44.21 | 902847 | 19.94 | 18 |

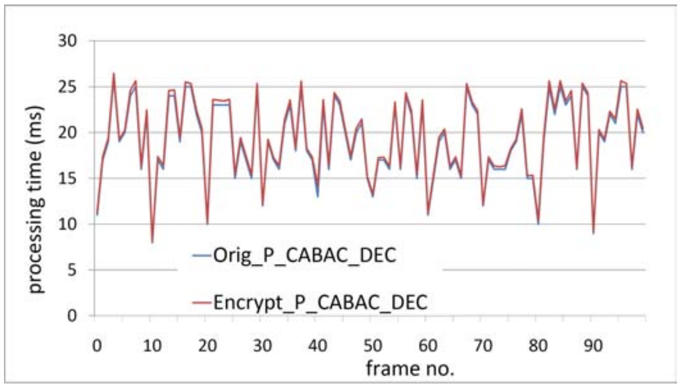
TABLE I: Analysis of ES for SE for different benchmark video sequences at QP value 18.

| QP | SE-CAVLC PSNR (dB) | SE-CAVLC Total Size (Bytes) | SE-CAVLC *ES* (%) | SE-CAVLC Avg. *ES* Bits/MB. | SE-CABAC PSNR (dB) | SE-CABAC Total Size Bytes | SE-CABAC *ES* (%) | SE-CABAC Avg. *ES* Bits/MB. |
|---|---|---|---|---|---|---|---|---|
| 12 | 50.07 | 1260001 | 28.55 | 36 | 50.05 | 1257024 | 19.97 | 25 |
| 18 | 44.38 | 813195 | 22.76 | 19 | 44.36 | 806063 | 18.72 | 15 |
| 24 | 39.43 | 478496 | 17.13 | 8 | 39.42 | 464794 | 17.61 | 8 |
| 30 | 35.08 | 268012 | 13.24 | 4 | 35.08 | 255287 | 15.65 | 4 |
| 36 | 31.04 | 145736 | 9.88 | 1 | 31.06 | 134143 | 12.22 | 2 |
| 42 | 27.23 | 88333 | 6.70 | 1 | 27.35 | 70616 | 9.46 | 1 |

TABLE II: Analysis of ES for SE over whole range of QP values for *foreman* video sequence.



(a)



(b)

Fig. 13: Framewise time taken by SE-CABAC of *foreman* video sequence for I+P frames at QP value 18 with *intra period* 10 during: a) Encoding, b) Decoding.

Table III gives a detailed overview of the required processing power for I and I+P video sequences at QP value 18. *intra period* has been set 10 for I+P video sequences. One can observe that increase in computation time for encoder is less than 0.4% for both of SE-CAVLC and SE-CABAC while it is below than 3% for decoder for I+P sequence.

Fig. 13.a and 13.b show the framewise analysis of increase in processing power for SE-CABAC at QP value 18 for *foreman*. For experimentation, 2.1 GHz Intel Core 2 Duo T8100 machine with 3072 MB RAM has been used. For I+P sequence encoding of 100 frames with *intra period* 10, it took 4372.5 seconds and 4381.3 seconds for CABAC and SE-CABAC respectively. While it took 2.005 seconds and 2.045 seconds for CABAC and SE-CABAC decoding. It is a negligible increase in processing power and can be managed well even by hand-held devices. It is important to note that increase is processing power of SE-CABAC is less than SE-CAVLC owing to two reasons. First, ES of SE-CABAC is lesser than that of SE-CAVLC as shown in Table I and Table II. Second, CABAC takes lot more processing power than CAVLC. So increase in processing power because of encryption will be lower in terms of percentage. Thus, SE-CAVLC and SE-CABAC is possible in real-time along with compression.

## *B. PSNR and Quality of SE-CAVLC & SE-CABAC for I Frames and I+P Frames*

Peak signal to noise ratio (PSNR) is widely used objective video quality metric. However, it does not perfectly correlate with a perceived visual quality due to non-linear behavior of human visual system. Structural similarity index (SSIM) [33] takes into account the structural distortion measurement, since human vision system is highly specialized in extracting structural information from the viewing field. SSIM has a better correlation to the subjective impression. SSIM ranges from $-1$ to $1$. SSIM is $1$ when both the images are the same.
To present the visual protection of encrypted video sequences, PSNR and SSIM of I and I+P frames are presented.

*1) I Frames:* To demonstrate the efficiency of our proposed scheme, we have compressed 100 I frames of each sequence at 30 fps. Fig. 14 and Fig. 15 show the encrypted first frame of *foreman* video sequence at different QP values for SE-CAVLC and SE-CABAC respectively. In H.264/AVC, blocks on the top array are predicted only from left while blocks on left are always predicted from top. Owing to this prediction, a band having width of 8 pixels at top of video frames can be observed for both of SE-CAVLC and SE-CABAC while this band has width of 4 pixels on left of video frames as shown in Fig. 14 and Fig. 15. The average PSNR values of *foreman* is given in Table IV over whole QP range. It is also compared with the PSNR obtained for the same video sequence without encryption. In Table IV we present PSNR of original video only for CAVLC. PSNR for CABAC is very much similar as presented in Table I. One can note that whatever is the QP value, the quality of the encrypted video remains in the same lower range.

Table V compares the average PSNR of 100 I frames of all benchmark video sequences at QP value 18 without encryption and with SE. Average PSNR value of *luma* for all the sequences at QP value 18 is $9.49$ $dB$ for SE-CAVLC and $9.80$ $dB$ for SE-CABAC. It confirms that this algorithm works well for various combinations of motion, texture and objects for I frames. It is also evident in frame-wise PSNR of *luma* of I frames of *foreman* video sequence as shown in Fig. 16.
Table VI contains the experimental results of SE of 100 I frames for SD resolution. Here, Average PSNR value of *luma* is $9.82$ $dB$ for SE-CAVLC and $9.83$ $dB$ for SE-CABAC, which is almost the same as that of QCIF resolution. It is evident that this algorithm is capable to encrypt high quality information at all resolutions. For the rest of the section, we present analysis for QCIF resolution only, since more benchmark video sequences are available in this resolution.

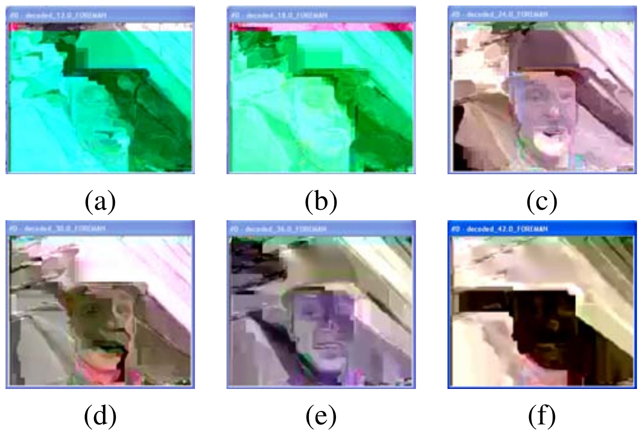


Fig. 14: Decoding of SE-CAVLC frame #1 of *foreman* video sequence with QP value equal to: a) 12, b) 18, c) 24, d) 30 e) 36, f) 42.

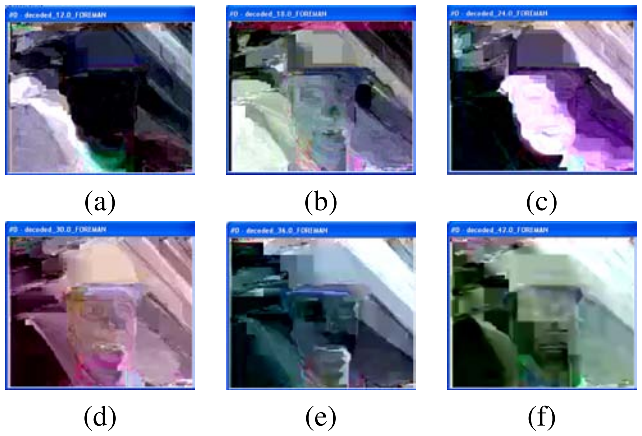


Fig. 15: Decoding of SE-CABAC frame #1 of *foreman* video sequence with QP value equal to: a) 12, b) 18, c) 24, d) 30, e) 36, f) 42.

Table VII shows the SSIM values of *luma* of benchmark video sequences without encryption and with SE. Results verify the proposed scheme has distorted the structural information present in the original video. Average SSIM value of video sequences without encryption is $0.993$, while it is $0.164$ and $0.180$ for SE-CAVLC and SE-CABAC respectively. Fig. 17 shows the frame-wise SSIM of *luma* of *foreman* video sequence for I frames. It is important to note SSIM value of complex video sequences is less than that of simple video sequences.

*2) I+P Frames:* Video data normally consists of an I frame and a trail of P frames. I frames are inserted periodically to restrict the drift because of lossy compression and rounding errors. In these experiments, *intra period* is set at 10 in a sequence of 100 frames. Results shown in Table VIII verify the effectiveness of our scheme over the

| | PSNR (Y) (dB) | | | PSNR (U) (dB) | | | PSNR (V) (dB) | | |
|---|---|---|---|---|---|---|---|---|---|
| QP | ORIG | SE-CAVLC | SE-CABAC | ORIG | SE-CAVLC | SE-CABAC | ORIG | SE-CAVLC | SE-CABAC |
| 12 | 50.07 | 8.61 | 8.43 | 50.00 | 19.78 | 24.09 | 50.79 | 9.57 | 22.58 |
| 18 | 44.38 | 8.67 | 8.58 | 45.68 | 24.14 | 24.40 | 47.57 | 10.16 | 22.10 |
| 24 | 39.43 | 8.71 | 8.72 | 41.93 | 26.39 | 24.35 | 44.19 | 24.91 | 22.84 |
| 30 | 35.08 | 9.43 | 8.69 | 39.75 | 27.45 | 24.58 | 41.41 | 25.36 | 23.64 |
| 36 | 31.04 | 9.37 | 8.53 | 37.74 | 28.12 | 24.93 | 38.62 | 24.78 | 23.16 |
| 42 | 27.23 | 9.45 | 8.67 | 36.23 | 25.51 | 24.91 | 36.86 | 24.59 | 24.02 |

TABLE IV: PSNR comparison for I frames without encryption and with SE for *foreman* at different QP values.

| | PSNR (Y) (dB) | | | PSNR (U) (dB) | | | PSNR (V) (dB) | | |
|---|---|---|---|---|---|---|---|---|---|
| Seq. | ORIG | SE-CAVLC | SE-CABAC | ORIG | SE-CAVLC | SE-CABAC | ORIG | SE-CAVLC | SE-CABAC |
| bus | 44.25 | 7.90 | 8.18 | 45.22 | 26.82 | 24.95 | 46.55 | 26.65 | 27.25 |
| city | 44.29 | 10.90 | 11.23 | 45.84 | 31.89 | 30.27 | 46.83 | 33.47 | 31.80 |
| crew | 44.82 | 8.96 | 9.90 | 45.84 | 23.99 | 23.45 | 45.70 | 19.74 | 19.79 |
| football | 44.61 | 11.48 | 11.49 | 45.77 | 14.85 | 14.39 | 46.05 | 24.28 | 23.59 |
| foreman | 44.38 | 8.67 | 8.58 | 45.68 | 24.14 | 24.40 | 47.57 | 10.16 | 22.10 |
| harbour | 44.10 | 9.25 | 9.50 | 45.61 | 27.07 | 24.61 | 46.67 | 33.25 | 31.31 |
| ice | 46.47 | 10.59 | 10.40 | 48.81 | 24.26 | 25.58 | 49.28 | 16.86 | 20.39 |
| mobile | 44.44 | 8.32 | 8.29 | 44.15 | 10.44 | 13.08 | 44.06 | 9.58 | 10.97 |
| soccer | 44.27 | 9.34 | 10.61 | 46.62 | 22.10 | 19.73 | 47.93 | 28.21 | 24.41 |
| avg. | 44.63 | 9.49 | 9.80 | 45.95 | 22.84 | 22.27 | 46.74 | 22.47 | 23.51 |

TABLE V: PSNR comparison for I frames without encryption and with SE at QP value 18.

| | PSNR (Y) (dB) | | | PSNR (U) (dB) | | | PSNR (V) (dB) | | |
|---|---|---|---|---|---|---|---|---|---|
| Seq. | ORIG | SE-CAVLC | SE-CABAC | ORIG | SE-CAVLC | SE-CABAC | ORIG | SE-CAVLC | SE-CABAC |
| city | 44.65 | 9.94 | 10.12 | 47.82 | 27.34 | 26.20 | 49.07 | 31.37 | 29.92 |
| crew | 45.15 | 9.16 | 9.08 | 46.56 | 24.52 | 22.80 | 47.74 | 20.14 | 19.97 |
| harbour | 44.54 | 9.35 | 9.37 | 47.48 | 22.91 | 22.92 | 48.73 | 28.79 | 26.81 |
| ice | 46.17 | 10.67 | 10.38 | 51.50 | 27.79 | 27.72 | 52.01 | 25.04 | 26.09 |
| soccer | 45.12 | 9.96 | 10.19 | 47.68 | 18.36 | 18.02 | 49.21 | 26.68 | 24.08 |
| avg. | 45.13 | 9.82 | 9.83 | 48.21 | 24.18 | 23.53 | 49.35 | 26.40 | 25.37 |

TABLE VI: PSNR comparison for I frames without encryption and with SE at QP value 18 (SD resolution).

| Seq. | ORIG-CAVLC | SE-CAVLC | ORIG-CABAC | SE-CABAC |
|---|---|---|---|---|
| bus | 0.995 | 0.069 | 0.994 | 0.064 |
| city | 0.994 | 0.115 | 0.994 | 0.093 |
| crew | 0.991 | 0.184 | 0.991 | 0.153 |
| football | 0.991 | 0.219 | 0.991 | 0.184 |
| foreman | 0.990 | 0.198 | 0.990 | 0.165 |
| harbour | 0.998 | 0.047 | 0.998 | 0.038 |
| ice | 0.990 | 0.419 | 0.990 | 0.398 |
| mobile | 0.998 | 0.040 | 0.998 | 0.356 |
| soccer | 0.988 | 0.185 | 0.988 | 0.171 |
| avg | 0.993 | 0.164 | 0.993 | 0.180 |

TABLE VII: SSIM comparison of *luma* of I frames without encryption and with SE at QP value 18.

| | PSNR (Y) (dB) | | | PSNR (U) (dB) | | | PSNR (V) (dB) | | |
|---|---|---|---|---|---|---|---|---|---|
| QP | ORIG | SE-CAVLC | SE-CABAC | ORIG | SE-CAVLC | SE-CABAC | ORIG | SE-CAVLC | SE-CABAC |
| 12 | 49.55 | 8.73 | 8.11 | 49.89 | 18.35 | 22.98 | 50.63 | 10.41 | 21.63 |
| 18 | 43.93 | 9.14 | 10.44 | 45.53 | 23.56 | 23.87 | 47.56 | 8.03 | 23.19 |
| 24 | 38.92 | 9.60 | 9.72 | 42.04 | 26.93 | 24.87 | 44.27 | 25.77 | 24.98 |
| 30 | 34.60 | 9.24 | 9.25 | 39.84 | 28.61 | 24.95 | 41.54 | 26.63 | 24.03 |
| 36 | 30.72 | 10.09 | 8.19 | 37.91 | 28.45 | 24.28 | 38.75 | 22.78 | 23.36 |
| 42 | 26.95 | 9.44 | 8.64 | 36.30 | 26.46 | 26.82 | 36.92 | 25.60 | 24.65 |

TABLE VIII: PSNR comparison for I+P frames without encryption and with SE for *foreman* at different QP values.

| | PSNR (Y) (dB) | | | PSNR (U) (dB) | | | PSNR (V) (dB) | | |
|---|---|---|---|---|---|---|---|---|---|
| Seq. | ORIG | SE-CAVLC | SE-CABAC | ORIG | SE-CAVLC | SE-CABAC | ORIG | SE-CAVLC | SE-CABAC |
| bus | 43.73 | 7.58 | 7.72 | 45.10 | 27.15 | 25.42 | 46.42 | 24.73 | 27.01 |
| city | 43.81 | 11.42 | 11.14 | 45.73 | 32.47 | 30.16 | 46.76 | 32.53 | 31.66 |
| crew | 44.46 | 8.97 | 10.00 | 45.81 | 25.09 | 21.98 | 45.73 | 19.63 | 20.18 |
| football | 44.16 | 12.13 | 11.28 | 45.72 | 14.31 | 14.58 | 46.06 | 24.77 | 24.27 |
| foreman | 43.93 | 9.14 | 10.44 | 45.53 | 23.56 | 23.87 | 47.56 | 8.03 | 23.19 |
| harbour | 43.71 | 9.46 | 9.78 | 45.45 | 24.53 | 22.93 | 46.58 | 33.87 | 31.67 |
| ice | 46.14 | 10.93 | 10.38 | 48.61 | 23.63 | 25.29 | 49.14 | 19.17 | 19.71 |
| mobile | 43.85 | 8.44 | 8.84 | 44.16 | 10.09 | 12.48 | 44.07 | 9.61 | 11.85 |
| soccer | 43.56 | 9.65 | 10.56 | 46.47 | 21.83 | 20.76 | 47.76 | 27.40 | 22.24 |
| avg. | 44.15 | 9.75 | 10.02 | 45.84 | 22.52 | 21.94 | 46.68 | 22.19 | 23.53 |

TABLE IX: Comparison of PSNR without encryption and with SE for I+P frames at QP value 18.

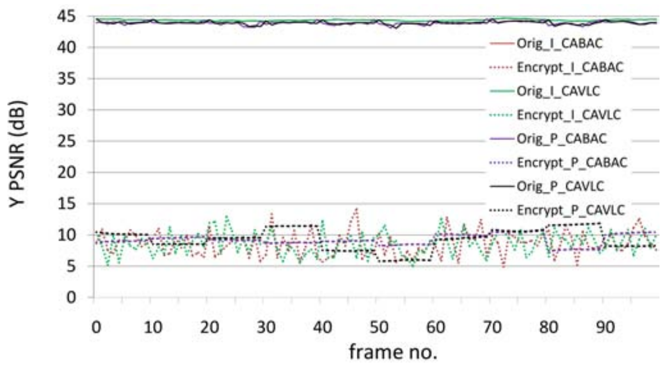


Fig. 16: Framewise PSNR of I and I+P frames for *foreman* for SE-CAVLC and SE-CABAC at QP value 18.

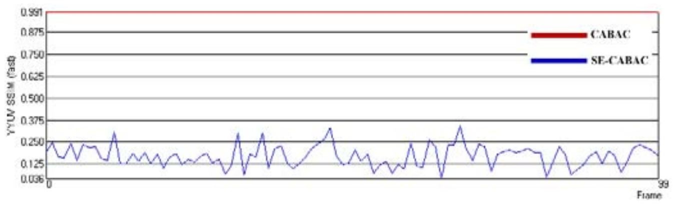


Fig. 17: Framewise SSIM of I frames for *foreman* for SE-CABAC at QP value 18.

whole range of QP values for *foreman* video sequence. Table IX verifies the performance of our algorithm for all video sequences for I+P frames at QP value 18. Average PSNR of *luma* for all the sequences is 9.75 $dB$ and 10.02 $dB$ for SE-CAVLC and SE-CABAC respectively. Fig. 16 shows the frame-wise PSNR of *luma* of *foreman* video sequence for I+P. Here PSNR of SE-CAVLC and SE-CABAC remains almost the same for sequence of P frames and changes at every I frame, thus producing a staircase graph. SSIM quality metric has very low values and is not given here for the sake of brevity.

## C. Security Analysis

*1) Analysis of entropy and local standard deviation:* The security of the encrypted image can be measured by considering the variations (local or global) in the protected image. Entropy is a statistical measure of randomness or disorder of a system which is mostly used to characterize the texture in the input images. Considering this, the information content of image can be measured with the entropy $H(X)$ and local standard deviation $\sigma(j)$. If an image has $2^k$ gray levels $\alpha_i$ with $0 \leq i \leq 2^k$ and the probability of gray level $\alpha_i$ is $P(\alpha_i)$, and without considering the correlation of gray levels, the $1^{st}$ order entropy $H(X)$ is defined as:

$$H(X) = -\sum_{i=0}^{2^k-1} P(\alpha_i) log_2(P(\alpha_i)). \tag{4}$$

If the probability of each gray level in the image is $P(\alpha_i) = \frac{1}{2^k}$, then the encryption of such image is robust against statistical attacks of $1^{st}$ order, and thus $H(X) = log_2(2^k) = k$ bits/pixel. In the image the information redundancy $r$ is defined as:

$$r = k - H(X). \tag{5}$$

Similarly the local standard deviation $\sigma(j)$ for each pixel $p(j)$ taking account of its neighbors to calculate the local mean $\overline{p(j)}$, is given as:

$$\sigma(j) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(p(i) - \overline{p(j)})}, \tag{6}$$

where $m$ is the size of the pixel block to calculate the local mean and standard deviation, and $0 \leq j < M$, if M is the image size.
In case of full encryption, entropy $H(X)$ is maximized with high values of local standard deviation. But in case of SE-CAVLC and SE-CABAC, the video frame is transformed to flat regions with blocking artifacts as depicted in Fig. 14 and Fig. 15. It is generally owing to variation in pixel values at MB boundaries. For

all the benchmark sequences, the average information redundancy $r$ for SE-CAVLC and SE-CABAC sequences is $0.94$ and $0.55$ respectively, while it is $1.11$ for all the original sequences. Despite the fact that SE-CAVLC and SE-CABAC transform the video frames into flat region, the entropy of the encrypted video sequences from equation (4) is higher as compared to original sequences.

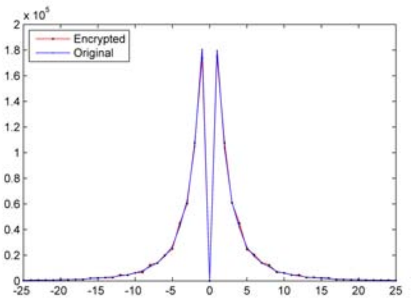
These flat regions are because of two reasons. Firstly, flat regions are due to the fact that prediction is performed from edge pixels of neighboring MBs. Secondly, pixels have either with very high value(bright video frame) or very low value (dark video frame) in SE video frame. This is owing to the fact that during reconstruction pixel value are clipped to $255$ if they are greater than it and to $0$ if they are below this lower range. So if many pixels have value beyond the upper or lower range, all of them will be clipped to the same value, thus creating a flat region which is either dark or bright. Based on this analysis, the statistical characteristics of SE-CAVLC and SE-CABAC bitstreams vary from full encryption systems. Fig. 18.a, 18.b and 18.c show the histogram of the original and the encrypted NZs for the *foreman* video sequence using for SE-CAVLC. Here we can see that SE has given an effect of staircase because of treating the coefficients with equal probability in the same interval.

From equation (6), we also analyzed the local standard deviation $\sigma$ for each pixel while taking into account its neighbors. In Table X, the mean local standard deviation for *foreman* sequence at different QP values is given. For all benchmark video sequences, the mean local standard deviation of *luma* equals to $69.15$ and $61.48$ for the SE-CAVLC and SE-CABAC bitstreams respectively, where the mean local standard deviation is less than $10$ gray levels for the original benchmark sequences. One can note that local standard deviation of encrypted sequences is higher than original sequences.

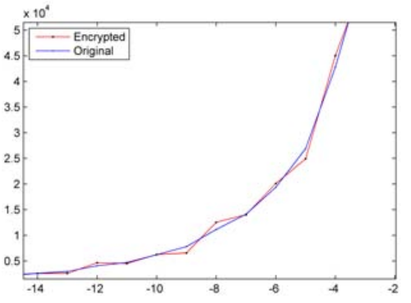
| | CAVLC | | CABAC | |
|---|---|---|---|---|
| QP | ORIG | SE-CAVLC | ORIG | SE-CABAC |
| 12 | 6.75 | 71.49 | 7.02 | 69.69 |
| 18 | 7.21 | 73.23 | 7.53 | 59.97 |
| 24 | 8.57 | 91.98 | 8.63 | 84.55 |
| 30 | 6.35 | 35.99 | 6.71 | 57.87 |
| 36 | 6.90 | 47.42 | 6.93 | 68.04 |
| 42 | 7.91 | 75.26 | 8.11 | 71.17 |

TABLE X: Standard deviation for SE of *foreman* video sequence at different QP values.

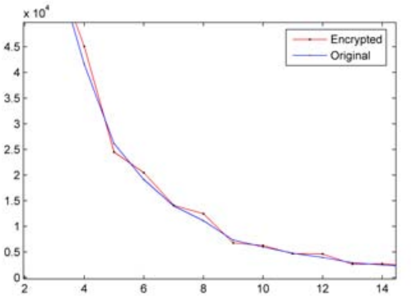
*2) Correlation of adjacent pixels:* Visual data is highly correlated i.e. pixels values are highly probable to repeat in horizontal, vertical and diagonal directions. A correlation of a pixel with its neighboring pixel is then



(a)



(b)



(c)

Fig. 18: Histograms of original and SE-CAVLC NZs: a) Complete graph, b) Zoomed graph of negative x-axis, c) Zoomed graph of positive x-axis.

given by a tuple $(x_i, y_i)$ where $y_i$ is the adjacent pixel of $x_i$. Since there is always three directions in images *i.e.* horizontal, vertical and diagonal, so we can define correlation direction between any two adjacent pixels as:

$$corr_{(x,y)} = \frac{1}{n-1}\sum_{0}^{n}(\frac{x_i - \overline{x_i}}{\sigma_x})(\frac{y_i - \overline{y_i}}{\sigma_y}), \qquad (7)$$

where $n$ represents the total number of tuples $(x_i, y_i)$, $\overline{x_i}$ and $\overline{y_i}$ represent the local mean and $\sigma_x$ and $\sigma_y$ represent the local standard deviation respectively.

Owing to the flat regions in SE-CAVLC and SE-CABAC video sequences, the correlation values in these sequences will be higher as compared to original image which contain texture and edges. For all the benchmark sequences, the average horizontal correlation coefficient is 0.88 and 0.87 for the SE-CAVLC and SE-CABAC respectively, while it is 0.80 for the original sequences.

*3) Key sensitivity test:* Robustness against cryptanalyst can be improved if the cryptosystem is highly sensitive towards the key. The more the visual data is sensitive towards the key, the more we would have data randomness. For this purpose, a key sensitivity test is assumed where we pick one key and then apply the proposed technique for encryption and then make a one bit change in the key and decode the bitstream. Numerical results show that the proposed technique is highly sensitive towards the key change, that is, a different version of encrypted video sequence is produced when the keys are changed, as shown in Fig. 19. PSNR of *luma* of decrypted frames with 1-bit different key is 10.39 dB and 8.31 dB for SE-CAVLC and SE-CABAC as shown in Table XI. It lies in the same lower range as decoded frames without decryption.

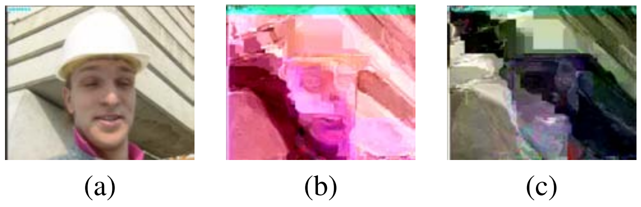


Fig. 19: Key sensitivity test for encrypted frame #1 of *foreman* video sequence for QP value 18. Encrypted frames are decrypted and decoded with: a) original key, b) 1-bit different key(SE-CAVLC), c) 1-bit different key(SE-CABAC).

*4) Removal of encrypted data attack:* In another experiment we have replaced the encrypted bits with constant values in order to measure the strength of SE-CAVLC and SE-CABAC proposed method as described in [27]. Here we have used frame #1 of *foreman* video sequence with QP value 24. Fig. 20 shows both encrypted and attacked video frames for SE-CAVLC and SE-CABAC. For example, Fig. 20.a shows SE-CAVLC video frame with $PSNR = 10.01\ dB$ for *luma*. If we set the encrypted bits of all NZs to zero, we get the video frame illustrated in Fig. 20.b with *luma* $PSNR = 8.87\ dB$. Similarly, Fig. 20.c shows SE-CABAC video frame having $PSNR = 8.20\ dB$ while the attacked SE-CABAC video frame has $PSNR = 7.72\ dB$ as shown in Fig. 20.d.

## D. Comparative evaluation

For the sake of comparative evaluation of our scheme, we have compared it with six other recent techniques, which include scrambling [9], NAL unit encryption [14],

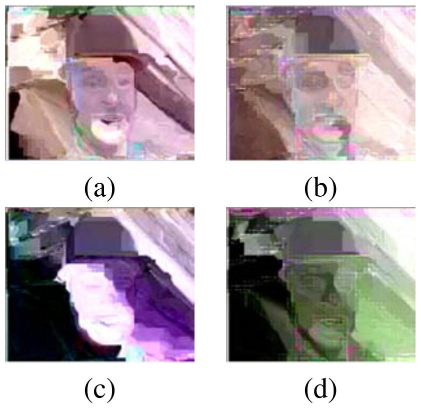


Fig. 20: Attack in the selectively encrypted image by removing the encrypted data: a) SE-CAVLC encrypted image {Y, U, V} = {10.01, 26.86, 25.24} dB, b) SE-CAVLC attacked image {Y, U, V} = {8.87, 27.3, 26.3} dB, c) SE-CABAC encrypted image {Y, U, V} = {8.20, 17.95, 24.53} dB, d) SE-CABAC attacked image {Y, U, V} = {7.72, 28.6, 24.6} dB.

MB header encryption [16], reversible ROI encryption [5], I frame encryption [2] and multiple Huffman table permutation [36]. These techniques are different from each other in several aspects e.g. working domain (pixel, transform or bitstream) and encryption algorithm (pseud orandom permutation, stream cipher or AES). The comparison has been made based on several important characteristics of SE systems and is summarized in Table XII.
Encryption algorithm used in SE scheme is of vital importance for the security level. AES has the highest security among all the known ciphers and our proposed scheme utilizes AES. Among the recent techniques, AES has been used only in [2] but their SE scheme is very naive and encrypts only I frames.

Selective encryption should not result in increase of bitrate. For example, if a video for 3G wireless connection has bitrate of 384 kbps. Its encrypted version should have the same bitrate. Otherwise it cannot not be played back on 3G connection. Our scheme keeps the bitrate intact. It is in contrast to other schemes which either allow increase in bitrate [9], [5], [36], or use stream cipher for the sake of same bitrate [14], [16], thus compromising on the security of the system.

Format compliance is another important aspect for encrypted video data. Most of the schemes are not format complaint and their encrypted bitstreams cannot be decoded by reference decoder except SE schemes which work in pixel domain [5] and transform domain [9].

Our SE-CABAC scheme is the first format compliant technique which is for arithmetic coding based entropy

| | PSNR (Y) (dB) | PSNR (U) (dB) | PSNR (V) (dB) |
|---|---|---|---|
| Original key | 44.60 | 45.73 | 47.35 |
| SE-CAVLC (1-bit different key) | 10.39 | 24.46 | 14.02 |
| SE-CABAC (1-bit different key) | 8.31 | 25.13 | 24.82 |

TABLE XI: Key sensitivity test of SE-CAVLC and SE-CABAC encrypted video for frame #1 *foreman* video sequence for QP value 18.

coding module, while keeping the bitrate unchanged. Recent encryption techniques for arithmetic coding [13], [11] are not format complaint and require lot of processing power.

To summarize, our proposed schemes (SE-CAVLC and SE-CABAC) meet all the requirements of an integrated compression-encryption systems. Our proposed system is fully compliant to H.264/AVC decoder, with no change in bitrate and has the security of AES cipher.

## V. Conclusion

In this paper, an efficient SE system has been proposed for H.264/AVC video codec for CAVLC and CABAC. The SE is performed in the entropy coding stage of the H.264/AVC using the AES encryption algorithm in the CFB mode. In this way the proposed encryption method does not affect the bitrate and the H.264/AVC bitstream compliance. The SE is performed in CAVLC *codewords* and CABAC *binstrings* such that they remain a valid *codewords/binstrings* thereafter having exactly the same length. Experimental analysis has been presented for I and P frames. The proposed scheme can be used for B frames without any modification, since B frames are also *inter* frames but have bidirectional prediction.
The proposed method has the advantage of being suitable for streaming over heterogeneous networks because of no change in bitrate. The experiments have shown that we can achieve the desired level of encryption, while maintaining the full bitstream compliance, under a minimal set of computational requirements. The presented security analysis confirms a sufficient security level for multimedia applications in the context of SE. The proposed system can be extended for ROI specific video protection [26] for video surveillance and can be applied to medical video transmission [24].

## VI. Acknowledgment

This work is in part supported by the VOODDO project (2008-2011) which is a French national project of ANR *Agence Nationale de la Recherche* and the region of Languedoc Roussillon, France.

## References

[1] "Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 / ISO/IEC 14496-10 AVC)," Joint Video Team (JVT), Doc. JVT-G050, Tech. Rep., March 2003.

[2] M. Abomhara, O. Zakaria, O. Khalifa, A. Zaiden, and B. Zaiden, "Enhancing Selective Encryption for H.264/AVC Using Advanced Encryption Standard," *International Journal of Computer and Electrical Engineering*, vol. 2, no. 2, pp. 223–229, 2010.

[3] M. Bellare, T. Ristenpart, P. Rogaway, and T. Stegers, "Format-Preserving Encryption," in *in Proc. 16th Annual International Workshop on Selected Areas in Cryptography*, Calgary, Canada, 2009, pp. 295–312.

[4] G. Bjontegaard and K. Lillevold, "Context-Adaptive VLC Coding of Coefficients," in *JVT Document JVT-C028*, Fairfax, VA, May 2002.

[5] P. Carrillo, H. Kalva, and S. Magliveras, "Compression Independent Reversible Encryption for Privacy in Video Surveillance," *EURASIP Journal on Information Security*, vol. 2009, p. 13, 2009.

[6] H. Cheng and X. Li, "Partial Encryption of Compressed Images and Videos," *IEEE Transactions on Signal Processing*, vol. 48, no. 8, pp. 2439–2445, Aug. 2000.

[7] J. Daemen and V. Rijmen, "AES Proposal: The Rijndael Block Cipher," Proton World Int.l, Katholieke Universiteit Leuven, ESAT-COSIC, Belgium, Tech. Rep., 2002.

[8] M. V. Droogenbroeck and R. Benedett, "Techniques for a Selective Encryption of Uncompressed and Compressed Images," in *Proc. of Advanced Concepts for Intelligent Vision Systems (ACIVS) 2002, Ghent, Belgium*, Sept. 2002, pp. 90–97.

[9] F. Dufaux and T. Ebrahimi, "Scrambling for privacy protection in video surveillance systems," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 8, pp. 1168 –1174, aug. 2008.

[10] M. M. Fisch, H. Stgner, and A. Uhl, "Layered Encryption Techniques for DCT-Coded Visual Data," in *Proc.* $12^{th}$ *European Signal Processing Conference (EUSIPCO'04), Vienna, Austria*, Sep. 2004, pp. 821–824.

[11] M. Grangetto, E. Magli, and G. Olmo, "Multimedia Selective Encryption by Means of Randomized Arithmetic Coding," *IEEE Transactions on Multimedia*, vol. 8, no. 5, pp. 905–917, Oct. 2006.

[12] G. Jakimoski and K. Subbalakshmi, "Cryptanalysis of Some Multimedia Encryption Schemes," *IEEE Transactions on Multimedia*, vol. 10, no. 3, pp. 330–338, April 2008. [Online]. Available: http://dx.doi.org/10.1109/TMM.2008.917355

[13] W. Jiangtao, K. Hyungjin, and J. Villasenor, "Binary arithmetic coding with key-based interval splitting," *IEEE Signal Processing Letters*, vol. 13, no. 2, pp. 69–72, Feb. 2006.

[14] C. Li, X. Zhou, and Y. Zong, "NAL Level Encryption for Scalable Video Coding," *Lecture notes in Computer Science, Springer*, no. 5353, pp. 496–505, 2008.

[15] S. Lian, Z. Liu, Z. Ren, and Z. Wang, "Selective Video Encryption Based on Advanced Video Coding," *Lecture notes*

| Video Selective Encryption Scheme | Format compliant | Robust to transcoding | Domain | Bitrate increase | Compression independent | Encryption algorithm |
|---|---|---|---|---|---|---|
| Scrambling for privacy protection [9] | Yes | No | Transform | Yes | Yes | Pseudo random sign inversion |
| NAL unit encryption [14] | No | No | Bitstream | No | No | Stream Cipher |
| MB header data encryption [16] | No | No | Transform | No | No | Stream Cipher |
| Reversible encryption of ROI [5] | Yes | Yes | Pixel | Yes | Yes | Pseudo random pixel permutations |
| I frame encryption [2] | No | No | Bitstream | No | No | AES |
| Multiple Huffman tables [36] | No | No | Bitstream | Yes | No | Huffman Table permutations |
| Our scheme | Yes | No | Bitstream* | No | No | AES (CFB mode) |

* For SE-CAVLC, bitstream is encrypted, while for SE-CABAC, binstrings are encrypted as explained in Section III-B.

TABLE XII: Comparison of proposed scheme with other recent methods.

*in Computer Science, Springer-verlag*, no. 3768, pp. 281–290, 2005.

[16] S. Lian, Z. Liu, Z. Ren, and H. Wang, "Commutative Encryption and Watermarking in Video Compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 6, pp. 774–778, June 2007.

[17] E. Lin, A. Eskicioglu, R. Lagendijk, and E. Delp, "Advances in Digital Video Content Protection," *Proc. of the IEEE*, vol. 93, no. 1, pp. 171–183, Jan. 2005.

[18] T. Lookabaugh and D. Sicker, "Selective Encryption for Consumer Applications," *IEEE Communications Magazine*, vol. 42, no. 5, pp. 124–129, May 2004.

[19] R. Lukac and K. Plataniotis, "Bit-Level Based Secret Sharing for Image Encryption," *Pattern Recognition*, vol. 38, no. 5, pp. 767–772, May 2005.

[20] D. Marpe, H. Schwarz, and T. Wiegand, "Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 620–636, July 2003.

[21] K. Martin, R. Lukac, and K. Plataniotis, "Efficient Encryption of Wavelet-Based Coded Color Images," *Pattern Recognition*, vol. 38, no. 7, pp. 1111–1115, Jul. 2005.

[22] I. Moccagatta and K. Ratakonda, "A Performance Comparison of CABAC and VCL-Based Entropy Coders for SD and HD Sequences," Joint Video Team (JVT), Doc. JVT-E079r2, Tech. Rep., Oct. 2002.

[23] S. Ou, H. Chung, and W. Sung, "Improving the Compression and Encryption of Images Using FPGA-Based Cryptosystems," *Multimedia Tools and Applications*, vol. 28, no. 1, pp. 5–22, Jan 2006.

[24] W. Puech and J. Rodrigues, "A New Crypto-Watermarking Method for Medical Images Safe Transfer," in *Proc. 12th European Signal Processing Conference (EUSIPCO'04), Vienna, Austria*, 2004.

[25] J.-M. Rodrigues, W. Puech, and A. Bors, "A Selective Encryption for Heterogenous Color JPEG Images Based on VLC and AES Stream Cipher," in *Proc. European Conference on Colour in Graphics, Imaging and Vision (CGIV'06), Leeds, UK*, Jun. 2006, pp. 34–39.

[26] ——, "Selective Encryption of Human Skin in JPEG Images," in *Proc. IEEE Int. Conf. on Image Processing, Atlanta, USA*, Oct. 2006, pp. 1981–1984.

[27] A. Said, "Measuring the Strength of Partial Encryption Scheme," in *Proc. IEEE Int. Conf. on Image Processing, Genova, Italy*, vol. 2, 2005, pp. 1126–1129.

[28] B. Schneier, *Applied cryptography*. Wiley, New-York, USA, 1995.

[29] Z. Shahid, M. Chaumont, and W. Puech, "Fast Protection of H.264/AVC by Selective Encryption," in *SinFra 2009, Singaporean-French IPAL Symposium, Fusionopolis*, Singapore, 18-20 Feb. 2009.

[30] ——, "Fast Protection of H.264/AVC by Selective Encryption of CABAC for I & P frames," in *Proc. 17th European Signal Processing Conference (EUSIPCO'09)*, Glasgow, Scotland, Aug. 2009, pp. 2201–2205.

[31] D. R. Stinson, *Cryptography: Theory and Practice, (Discrete Mathematics and Its Applications)*. New York: Chapman & Hall/CRC Press, November 2005.

[32] L. Tang, "Methods for Encrypting and Decrypting MPEG Video Data Efficiently," in *Proc. ACM Multimedia*, vol. 3, New York, NY, USA, 1996, pp. 219–229.

[33] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," *IEEE Transactions on Image Processing*, vol. 13, pp. 600–612, 2004.

[34] J. Wen, M. Severa, W. Zeng, M. Luttrell, and W. Jin, "A Format-Compliant Configurable Encryption Framework for Access Control of Video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 6, pp. 545–557, Jun. 2002.

[35] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the h.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, July 2003.

[36] C.-P. Wu and C.-C. Kuo, "Design of Integrated Multimedia Compression and Encryption Systems," *IEEE Transactions on Multimedia*, vol. 7, pp. 828–839, 2005.

[37] K. Yabuta, H. Kitazawa, and T. Tanaka, "A New Concept of Security Camera Monitoring with Privacy Protection by Masking Moving Objects," in *Proc. Advances in Multimedia Information Processing*, vol. 1, no. LNCS 3767, 2005, pp. 831–842.

[38] W. Zeng and S. Lei, "Efficient Frequency Domain Selective Scrambling of Digital Video," *IEEE Transactions on Multimedia*, vol. 5, pp. 118–129, 2003.

[39] S. Ziauddin, I. U. Haq, and M. A. Khan, "Method and System for Fast Context based Adaptive Binary Arithmetic Coding," Patent US7 221 296, 2007.