Joint Entropy Coding and Encryption in AVS Video Codec

Zafar Shahid, Marc Chaumont, William Puech

Joint Entropy Coding and Encryption in AVS Video Codec

Z. Shahid, M. Chaumont and W. Puech
LIRMM Labs,
University of Montpellier II, France

1 Introduction

AVS is the state of the art video coding standard of China. Like H.264/AVC, a video frame is processed into blocks of 16x16 pixels, called macroblock (MB) in AVS. Each MB can be encoded as *intra* or *inter*. Spatial prediction is performed on blocks of 8x8 in *intra* frame, in contrast to 4x4 and 16x16 block size in H.264/AVC. It is less complex with only five modes for *luma* prediction, as compared to thirteen for *luma* prediction modes in H.264/AVC. Reference pixels, which are to be used for prediction, are first low pass filtered in some of the modes of AVS. In *inter* frame, it supports variable block size motion estimation up to 8x8 block, quarter pixel motion estimation and up to two reference frames. DCT has been replaced by Integer Cosine Transform (ICT) (Ma & Gao, 2006) in AVS. For quantization, a QP value ranges 0-63 with a period of approximate 8. AVS decoder complexity is further reduced by moving the inverse scaling from decoder to encoder module. AVS Part-2 supports several profiles. Two entropy coding modes are supported, namely Context-based 2D variable length coding (C2DVLC) in Jizhun profile and context-based binary arithmetic coding (CBAC) in Jiaqiang profile (Zhang et al., 2009).

After an introduction of AVS and its entropy coding, we present a proposition regarding joint entropy coding and selective encryption of AVS video codec for its C2DVLC entropy coding engine. The main difference between CAVLC of H.264/AVC and C2DVLC of AVS from real-time selective encryption point of view is that C2DVLC codewords are not contiguous and do not use the full code-space. In this case, C2DVLC codewords are mapped assigned indices, which are then encrypted. This scheme works fine if codewords are encrypted separately. For real time selective encryption using state of the art Advanced Encryption Standard (AES) algorithm in Cipher Feedback CFB) mode, a plaintext is prepared by concatenation of encrypt-able bits of codewords. In case of C2DVLC, the encrypted codewords may not be valid codewords because C2DVLC encrypt-able bits do not use a full code-space. In this chapter, we will present two schemes to overcome this limitation for real-time real-time selective encryption of C2DVLC.

This chapter is arranged as follows. We have presented introduction to C2DVLC of AVS in Section 2. The concept of selective and partial encryption is introduced in Section 3, while Section 4 contains a discussion on recent work for selective encryption of visual content. The proposed technique for joint entropy coding and encryption is presented in Section 5. At the end, concluding remarks regarding the whole chapter are presented in Section 6.

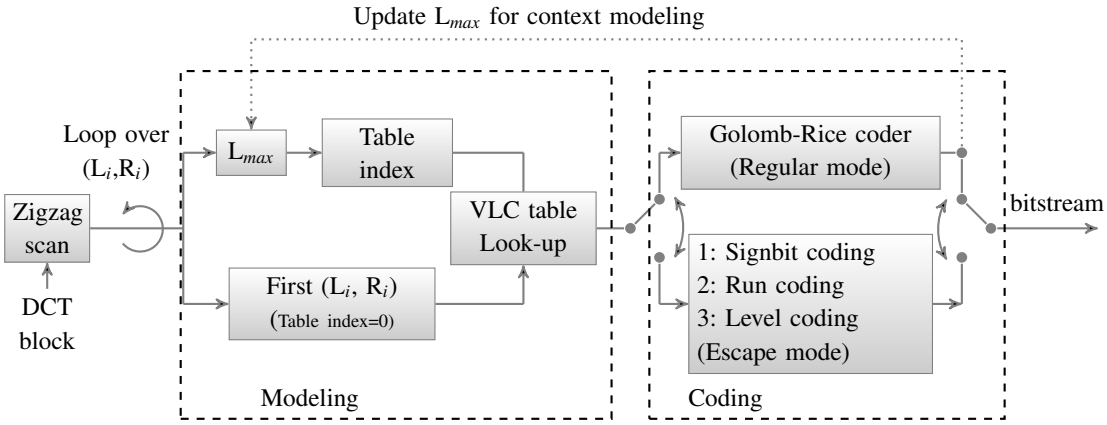


Figure 1: Block diagram of C2DVLC entropy coding module of AVS.

2 Context-based 2D variable length coding (C2DVLC) of AVS

Audio Video coding Standard (AVS) (Fan et al., 2004) is the state of the art video coding standard of China. It has slightly less performance but has much less complexity than H.264/AVC (H264, 2003). In comparison to H.264/AVC, AVS Part-2 Jizhun profile has about 3% efficiency loss as compared to H.264/AVC main profile in terms of bit saving on HD progressive-scan sequences (Wang et al., 2004). 8x8 transform coding, 8x8 spatial prediction, motion compensation up to 8x8 block, 8x8 in-loop deblocking filter and 2D variable length coding are major tools of AVS Part-2 which distinguishes it from H.264/AVC.

C2DVLC is an efficient context-based 2D-VLC entropy coder designed for coding 8x8 block-size transform coefficients. 2D-VLC means that a pair of run-level (L_i, R_i) is regarded as one event and jointly coded (Wang et al., 2006). Fig. 1 illustrates the working of C2DVLC by a flowgraph. A DCT block contains several transform coefficients which are transformed to (L_i, R_i) pairs after the zigzag scan. C2DVLC starts the coding in reverse order using 2D-VLC table with $TableIndex = 0$. Every table has certain range for (L_i, R_i) as shown in Fig. 2. If the current (L_i, R_i) lies in the range of current 2D-VLC table, it is encoded by *regular mode*. Otherwise *escape mode* is used.

In *regular mode*, the value of syntax element is firstly mapped to a non-negative integer *CodeNumber* using a table look-up operation. These *CodeNumbers* are then mapped to corresponding Exp-Golomb codewords. For (L_i, R_i) pairs having negative value for *level*, *CodeNumber* is incremented by 1. For example, for $(L_i, R_i) = (2, 1)$ *CodeNumber* is 11 and for $(L_i, R_i) = (-2, 1)$ *CodeNumber* is 12. Exp-Golomb codes have regular structures, which means that any non-negative *CodeNumber* can be mapped to a unique binary codeword using the regular code-constructing rule. Due to the regular codeword structure, the binary code for a given *CodeNumber* can be constructed in coding process without involving high computational complexity. In AVS, it is a valuable feature that resolves the problem of high memory requirement for multiple VLC tables. In *escape mode*, L_i and R_i are coded separately using Exp-Golomb codewords. R_i and sign of L_i are jointly coded. While for the magnitude of L_i, a prediction is first performed and then the prediction error is coded.

C2DVLC switches the 2D-VLC tables based on the maximum magnitude of the previously coded *levels*. Let L_{max} be the maximum magnitude of the previously coded *levels*. The *TableIndex* for coding of next (L_i, R_i) is updated if L_{max} is greater than the threshold of the current table as given below:

$$TableIndex = \quad j, \quad \text{if } (Th[j+1] > L_{max} \geq Th[j]) \tag{1}$$

with the threshold for each table given as:

$$Th[0\ldots7]\begin{cases}(0,\ 1,\ 2,\ 3,\ 5,\ 8,\ 11,\ \infty) & \textit{intra\_luma}\\ (0,\ 1,\ 2,\ 3,\ 4,\ 7,\ 10,\ \infty) & \textit{inter\_luma}\\ (0,\ 1,\ 2,\ 3,\ 5,\ \infty,\ \infty,\ \infty) & \textit{chroma}\end{cases} \tag{2}$$

This process is repeated for all the (L_i, R_i) pairs. At last, the EOB flag is coded to signal the end of block.

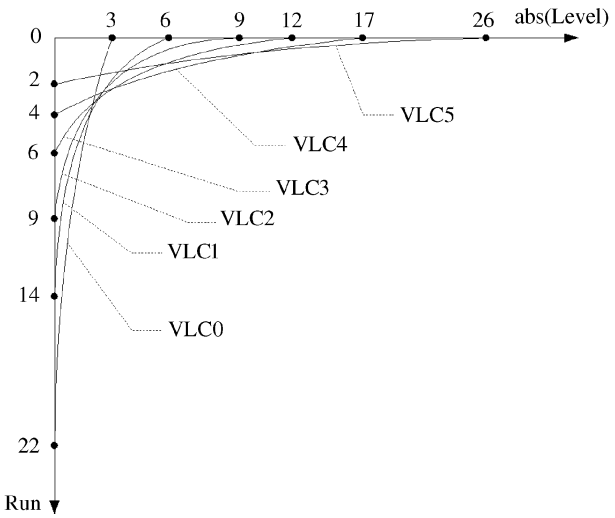


Figure 2: Limits of 2D-VLC tables of C2DVLC.

2D-VLC entropy coding has already been used in former video coding standards such as MPEG-2/4. But it has two main differences here. First, Huffman coding has been replaced by Exp-Golomb coding in AVS. Second, former video coding standards are not adaptive and use one single VLC table to code a certain type of transform blocks, e.g. one table for *intra* blocks, one table for *inter* blocks etc.

In AVS, 19 2D-VLC tables have been introduced for coding of residual coefficients and the memory requirement is only about 1k bytes. This method gives gain up to 0.23 dB compared to one-table-for-one-type-of-block coding method (Wang et al., 2004). For further details about C2DVLC, please refer to (Zhang et al., 2009).

Both C2DVLC of AVS and CAVLC of H.264/AVC are based on VLC. They are adaptive to the local statistics of DCT coefficients and coding efficiency of both of them is similar. But C2DVLC is substantially different to CAVLC in two aspects. First, Exp-Golomb coding is used **only** for coding of header syntax elements in CAVLC. On the other hand, all the syntax elements including transform coefficients are coded using Exp-Golomb coding in AVS. Second, transform coefficients are converted to *levels* and *runs*, which are coded **separately** using multiple VLC tables in CAVLC. While in C2DVLC, transform coefficients are first converted to (L_i, R_i) pairs. These pairs are mapped to *Codenumber* which is coded using Exp-Golomb codes in *regular mode*. In *escape mode*, L_i and R_i are coded separately using Exp-Golomb codes.

3 Selective and partial encryption of multimedia data

Digital rights management (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 depending on the application requirements. For example military and law enforcement applications require full encryption. Nevertheless, there is a large spectrum of applications that demand security on a lower level, as for example that ensured by selective encryption (SE) or partial encryption (PE). Such approaches reduce the computational requirements in networks with diverse client device capabilities (Cheng & Li, 2000).

Selective encryption encrypts some part of the data to get desired level of security and utilizes minimal computational resources. While in partial encryption, we encrypt only a portion of image (*e.g.*, ROI), rather than encryption of the whole image. SE and PE can be used to process and transmit the visual content in real-time. The security level of PE or SE is always lower when compared to the full encryption. On the other hand PE or SE decreases the data size to be encrypted and consequently requires lower computational time which is an important asset in wireless and portable multimedia systems. In this case we have a trade-off between the amount of encrypted data and the necessary computational resources.

Confidentiality is very important for low powered embedded systems such as for example smart phones. When considering image processing applications on such devices we should use minimal resources. The modern ciphers are usually too slow to be used for image and video processing in commercial low powered systems. The selective encryption (SE) can fulfill the application requirements without the computational overhead of the full encryption. In case of SE, only the minimum necessary data are ciphered. However, the security of SE is always lower when compared to that of the full encryption. Substantial computational reduction is one of the main advantages of SE and PE. The security of partially encrypted videos to attacks which exploit the information from non-encrypted bits together with the availability of side information has been studied in (Said, 2005).

Selective encryption can be format compliant and can leave important header information without encryption. Format compliance is a required feature in many applications such as rate adaptation for multimedia on heterogeneous networks (Li, 2001), DC image extraction for multimedia content searching (Yeo & Liu, 1995). It is also required for bandwidth adaptation (Wu et al., 2000), unequal error adaptation (Wang et al., 2000) and random access (Wen et al., 2002). Moreover, the relation between the quality and timeliness is very important. For example, a football match requires protection during the show to ensure revenue from the viewers, but the demand of security reduces or even vanishes over the following days.

Modern encryption algorithms including AES take lot of processing power. They are not suitable for encryption of multimedia mainly for two reasons. First, multimedia data is huge in size and full encryption demands lot of processing power. Second, full encryption will also encrypt the header information, which is critical for certain operations. This limitation gives rise to introduction of low-processing techniques which do not take lot of processing power but protects the visual data by degrading the quality *e.g.*, selective encryption (SE), partial encryption (PE) and soft encryption. AES in CFB mode is a suitable candidate for selective encryption, wherein it acts as a stream cipher. In Section 3.1, AES encryption cipher is explained. It is followed by its working modes in Section 3.2.

3.1 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 (Daemen & Rijmen, 2002). 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. The same number is 11, if either the block or the key is 192 bits long or if neither of them is longer than that. If either the block or the key is 256 bits long the number of rounds is 13. Given a sequence $\{X_1, ..., X_n\}$ of bit

plaintext blocks, each X_i is encrypted with the same secret key k producing the ciphertext blocks $\{Y_1, ..., Y_n\}$ as described in Fig. 3.

To encipher a data block X_i in AES, an AddRoundKey step is performed first by XORing a subkey with the block. The incoming data and the key are added together in the first AddRoundKey step. Afterwards, it follows the round operation. Each regular round operation involves four steps which are SubBytes, ShiftRows, MixColumns and AddRoundKey. In the SubBytes step, each byte of the block is replaced by its substitute in a substitution box (S-Box). In cryptography, an S-box is a basic component of symmetric key algorithms used to obscure the relationship between the plaintext and the ciphertext. The next one is the ShiftRows step where the rows are cyclically shifted over different offsets. The next step is the MixColumns, where each column is multiplied with a matrix over the Gallois Field, denoted as GF(2^8). The last step of the round operation is another AddRoundKey. It is a simple XOR with the actual data and the subkey for the current round. Before producing the final ciphered data Y_i, the AES performs an extra final routine that is composed of (SubBytes, ShiftRows and AddRoundKey) steps as shown in Fig. 3.

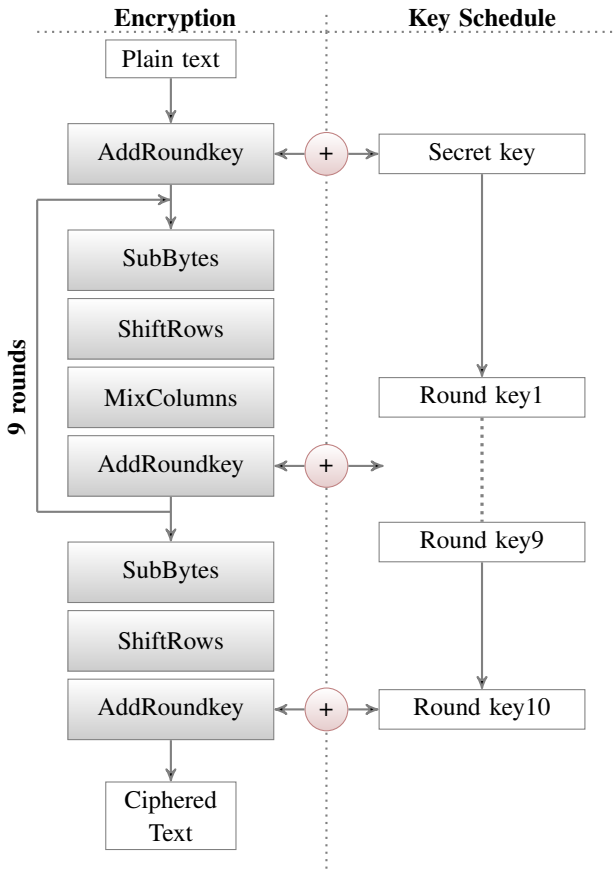


Figure 3: Block diagram of AES encryption algorithm containing 9 rounds of processing steps.

The process over the plaintext data X_i is independent of the process over the secret key, and is called KeySchedule. It is made up of two components: the KeyExpansion and the RoundKeySelection. The Expanded-Key is a linear array of 4-byte words and is given by $W[N_b \times (N_k + 1)]$, where N_b is the number of columns of the data block and N_k is the number of columns of the cipher key. The first N_k words contain the cipher key and all of

the other words are defined recursively. The KeyExpansion function depends on the value of N_k while the cipher key is expanded into an ExpandedKey.

3.2 Working modes for AES

AES cipher can support several cipher modes *e.g.*, ECB (electronic code book), CBC (cipher block chaining), OFB (output feedback), CFB (cipher feedback), CTR (Counter)(Stinson, 2005) and others. 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:

$$Y_i = E_k(X_i), \tag{3}$$

where $E_k(.)$ is the block encryption function. 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, Y_0 is substituted by the IV as shown in Fig. 4. The keystream element Z_i is then generated and the ciphertext block Y_i is produced as:

$$\begin{cases} Z_i &= E_k(Y_{i-1}), \text{for } i \geqslant 1 \\ Y_i &= X_i \oplus Z_i \end{cases}, \tag{4}$$

where $\oplus$ is the XOR operator.

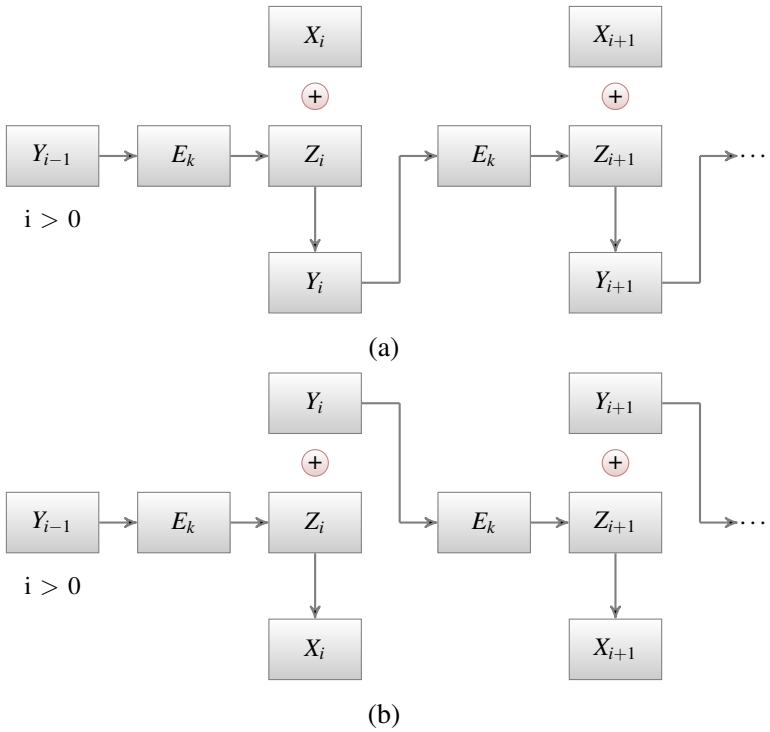


Figure 4: Cipher feedback mode of block cipher: (a) Encryption process, (b) Decryption process.

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 exactly that.

4 Prior work on SE and PE of image and video data

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 (Lookabaugh & Sicker, 2004). SE and 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 (Cheng & Li, 2000; Martin et al., 2005; Rodrigues et al., 2006).

The technical challenges posed by DRM systems are high and previous approaches have not entirely succeeded in tackling them (Lin et al., 2005). 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 (Yabuta et al., 2005).

Different encryption techniques including permutation, DES and AES have been used for SE of image and video in literature. Theoretically, video codec can be interrupted at any point to perform encryption. But the candidate domains for SE can be divided into five broad categories namely spatial, video codec structure, transform, entropy coding stage and bitstream as shown in Fig. 5. In this section, the state of the art of SE and PE techniques for image/video is presented in the following subsections.

4.1 Spatial domain encryption

Spatial domain is a straight forward place to apply generic encryption to multimedia as shown by stage 1 in Fig. 5. There is increase in bitrate for such methods because encryption increases the entropy of the visual data. The main advantage of these schemes is that they are independent of any compression standards and if they are robust against compression, they can also withstand video transcoding attacks. Spatial domain SE of H.264/AVC has been discussed in (Carrillo et al., 2009) wherein they do permutations of the pixels of MBs which are in ROI. The drawback of this scheme is that bitrate increases as the size of ROI increases. This is due to increase in entropy of ROI region.

Another technique which encrypts video pixels by using different SCAN patterns was proposed by Maniccam and Bourbakis (Maniccam & Bourbakis, 2004). They took difference of frames and encrypted the residual data. The drawback of this method is that the bitrate of video bitstream will rise by several times.

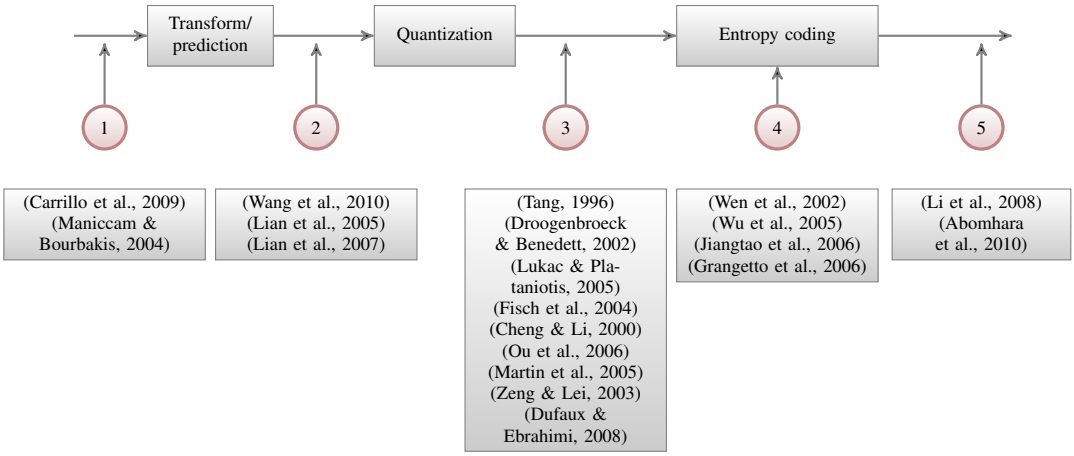


Figure 5: Encryption of video data before, during and after compression at different stages: (1) Spatial, (2) Video codec structure, (3) Transform domain, (4) Entropy coding stage, (5) Bitstream.

4.2 Encryption in the video codec structure

Instead of performing SE in video data (e.g. pixels, coefficients, entropy codes), some researchers have also proposed the encryption of header data. At stage 2 in Fig. 5, all of this header information is available for encryption.

In (Wang et al., 2010), Wang *et al.* have presented a SE scheme for H.264/AVC. They proposed to encrypt intra-prediction mode, motion vector difference, and quantization coefficients. A hierarchical key generation scheme is also proposed, in which the encryption keys are generated based on the cryptographic hash function. Since generated frame keys have been consistent with the corresponding frame numbers, frames are always synchronized in the decrypting process even if the frame loss occurs. Hence this scheme is secure against some frame synchronization attacks.

In (Lian et al., 2005), Lian *et al.* have performed SE 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 (Lian et al., 2007). 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 these techniques is that encrypted video bitstream will not be fully format compliant. If video frame headers are encrypted, the bitstream will be recognized as video bitstream but it will not be browse-able at frame level. If only MB header data are encrypted, the video bitstream cannot be parsed browse-able at MB level. In case the system level headers are encrypted. The bitstream is not format compliant at all.

4.3 Transform domain encryption

Video data can also be encrypted in transform domain as shown by stage 3 in Fig. 5. Several transform domain SE techniques have been proposed in the literature.

In (Tang, 1996), Tang proposed a technique called scan permutation applicable to DCT-based image and video codecs. This method provides a certain level of confidentiality, but it increases the overall bitrate. In (Droogenbroeck & Benedett, 2002), 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. In [zeng2003], Zeng and Lei proposed to shuffle quantized transform coefficients across DCT blocks, but within the same frequency band.

Combining PE and image/video compression using the set partitioning in hierarchical trees was used in (Cheng & Li, 2000). 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 (Lukac & Plataniotis, 2005).

The AES was applied on the Haar discrete wavelet transform compressed images in (Ou et al., 2006). The encryption of color images in the wavelet transform has been addressed in (Martin et al., 2005). In this approach the encryption is performed on the resulting wavelet code bits. Fisch *et al.* (Fisch et al., 2004) 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.

In (Zeng & Lei, 2003), 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 (Dufaux & Ebrahimi, 2008). It performs SE by pseudorandomly inverting sign of DCT coefficients in ROI.

4.4 Encryption in the entropy coding module

Encryption during the entropy coding module has been investigated by several authors. It is shown by stage 4 in Fig. 5.

The use of Huffman entropy coder as encryption cipher has been studied in the literature in (Wen et al., 2002; Wu et al., 2005). Entropy coding based SE of MPEG4 video standard has been studied in (Wen et al., 2002) 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 tradeoff has to be made among complexity, security and the bit overhead. In (Wu et al., 2005), Wu and Kuo have presented an encryption technique based on statistical models. They proposed to perform encryption 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 (Jakimoski & Subbalakshmi, 2008).

The conversion of arithmetic entropy coding module into encryption cipher has been proposed in (Jiangtao et al., 2006; Grangetto et al., 2006). In (Jiangtao et al., 2006), key-based interval splitting of arithmetic coding (KSAC) has been proposed 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 (Grangetto et al., 2006) 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 entropy coding based techniques is that encrypted bitstream is not format compliant. Moreover, arithmetic coding based techniques require lot of processing power.

4.5 Encryption in the video bitstream

Encryption of the compressed video is explored by few researchers. It is shown by stage 5 in Fig. 5. SE of H.264/AVC at network abstraction layer (NAL) has been proposed by (Li et al., 2008). Important NAL units namely instantaneous decoding refresh (IDR) picture, sequence parameter set (SPS), and picture parameter set (PPS) are encrypted with a stream cipher. This scheme is not format compliant and cannot be parsed even at

frame level. SE of H.264/AVC using block-based AES has been proposed in (Abomhara et al., 2010). In this scheme, encryption of I frame is performed, since P and B frame have no significance without I frames. The limitation of SE at this stage is that network level decisions (*e.g.*, unequal error protection, network adaptive bit allocation etc.) cannot be taken, since video header information is also encrypted.

5 Real-time selective encryption of AVS

In this Section, we present an algorithm for selective encryption of AVS, followed by its real-time implementation. It is performed in the C2DVLC module of video codec. For this purpose, AES cipher with the Cipher Feedback (CFB) mode is used on a subset of codewords. C2DVLC serves the purpose of encryption step without affecting the coding efficiency of AVS by keeping the bitrate unchanged, generating completely compliant bit stream and utilizing negligible computational power.

In contrast to SE-CAVLC of H.264/AVC, certain constraints have to be fulfilled for real-time format compliant SE of C2DVLC (SE-C2DVLC). The main differences between CAVLC of H.264/AVC and C2DVLC of AVS from selective encryption point of view are that C2DVLC codewords are not contiguous and they do not use the full code-space. SE of C2DVLC codewords is being presented in Section 5.1, while the proposed schemes for its real-time implementation are discussed in Section 5.2.

5.1 Selective encryption of C2DVLC (SE-C2DVLC))

We want SE-C2DVLC scheme which fulfills real-time constraints (same bitrate, format compliance), while utilizing minimal processing power. To keep the bitrate unchanged along with encrypted bitstream format compliance, we perform encryption of C2DVLC while fulfilling the following **constraints**:

- In the (L_i, R_i) pair, only L_i can be encrypted. R_i value must not be changed otherwise the bitstream will not be decodable.
- From equation (1), the encrypted symbol should be such that L_{max} remains in the same interval, thus selecting the same context for the next (L_i, R_i).
- For Exp-Golomb coding, the length of the encrypted codeword must be equal to that of original codeword.

Encryption of C2DVLC is not straight forward like that of CAVLC because of these constraints. In CAVLC, code-space is always full and we have specific bits which can be encrypted. But in case of C2DVLC, for *regular mode*, we can encrypt only the levels and their sign bits while taking into account the constraints described above.

(L_i, R_i) pair is mapped to *CodeNumber*, which is then coded using Exp-Golomb codeword, either in *regular* or *escape* mode. In *regular mode*, code-space is not full because of two major **limitations**:

- Non-consecutive *CodeNumbers* are assigned to consecutive levels.
- We do not have specific bits to be encrypted and the encryption space (ES) is not a power of 2.

In *escape mode*, we can encrypt the sign bit and suffix of the Exp-Golomb codeword. Here code-space is not guaranteed to be full because of second constraint (*i.e.* L_{max} *remains in the same interval*).

Fig. 6 encircles the functional blocks with constraints. It also shows the encryptable functional blocks.

To solve the limitations that Non-consecutive *CodeNumbers* are assigned to consecutive levels, C2DVLC codewords can be mapped to indices which are then encrypted. In this case, the encryption operation can be defined as:

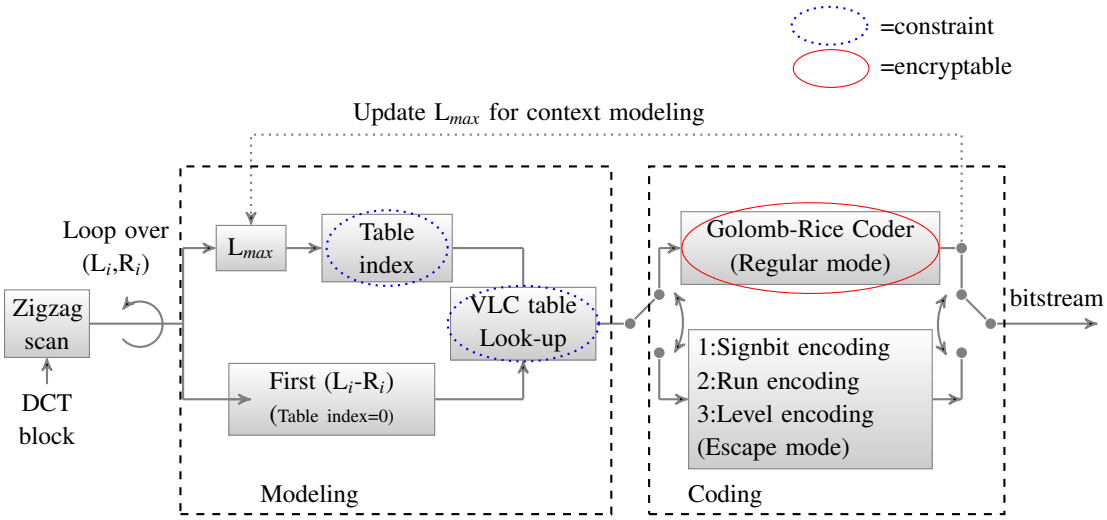


Figure 6: Block diagram of C2DVLC showing constraints and encrypt-able blocks. Suffixes are mapped to indices to fulfill first constraint of non-contiguous *CodeNumbers* for contiguous levels. Second constraint of non-full code-space is handled to have real-time implementation.

$$Y_i = Encrypt(X_i) = T^{-1}[\mathcal{E}(T(X_i)))], \tag{5}$$

where $\mathcal{E}(.)$ is a AES encryption function and $T(.)$ represents a bijective mapping between symbols with the same code-length and indices. The goal of this mapping is to pass indices to subsequent encryption functions. The corresponding decoding process will be:

$$X_i = Decrypt(Y_i) = T^{-1}[\mathcal{D}(T(Y_i)))], \tag{6}$$

where $\mathcal{D}(.)$ is a decryption functions corresponding to $\mathcal{E}(.)$.

For SE-C2DVLC, AES algorithm is used in Cipher Feedback (CFB) mode for encryption, similar to SE of H.264/AVC,. In this mode, AES is a stream cipher. Each ciphertext block Y_i is XORed with the incoming plaintext block X_{i+1} before being encrypted with the key k. For the first iteration, Y_0 is substituted by an initialization vector (IV).

For example, let us code current pair $(L_i, R_i) = (6, 0)$ with $TableIndex = 3$ for *intra_luma* mode as shown in Fig. 7. By encoding it with *regular mode* of C2DVLC, its *CodeNumber* will be 17 and its Exp-Golomb codeword will take 7 *bits*. Now let us examine the ES available for this (L_i, R_i) pair.

The first constraint is that only L_i can be encrypted in the (L_i, R_i) pair. So the ES consists of the levels which have valid codeword in $TableIndex = 3$ with $R_i = 0$. These are *CodeNumbers* {8, 0, 2, 4, 9, 11, 17, 21, 25, 33, 39, 45, 55} related to levels {0, 1,..., 12}. The second constraint is that the magnitude of the encrypted L_i should be within the interval that creates the same *TableIndex* for the next (L_i, R_i). From equation (2), we see that the current $L_i = 6$ will increase the *TableIndex* from 3 to 4 for the next (L_i, R_i) pair to be coded. So the encrypted L_i should also be in the same interval i. e. {5, 6, 7}. From *Table* with $TableIndex = 3$, the *CodeNumbers* for these levels are {11, 17, 21} for positive and {12, 18, 22} for negative sign. The third constraint implies that out of these *CodeNumbers*, only those make the ES which have the same Exp-Golomb codeword length as the original level (7 *bits*). Out of the 6 *CodeNumbers* which have been selected in the last step, five *CodeNumbers*

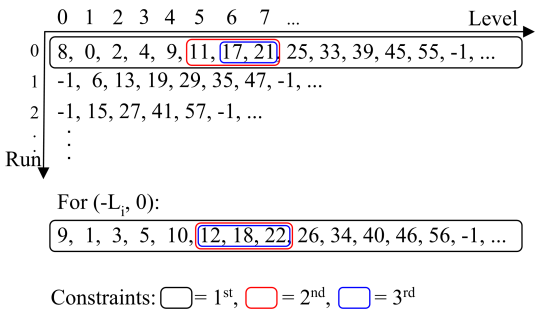


Figure 7: $(L_i, R_i) = (6,0)$ encryption in *regular mode* for *Table* with $TableIndex = 3$.

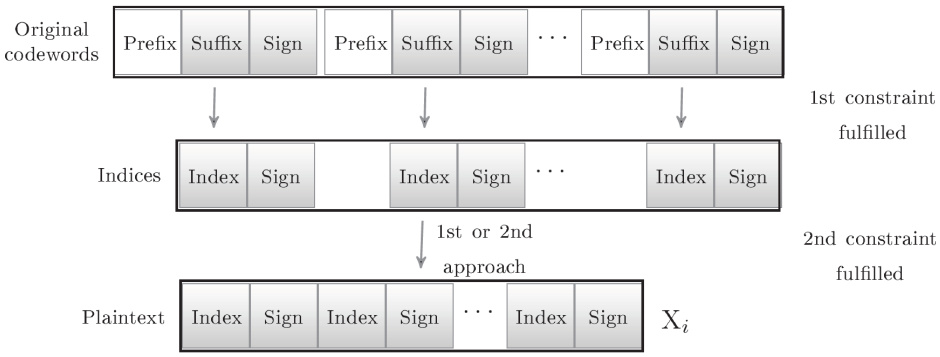


Figure 8: Construction of plaintext for AES cipher in CFB mode.

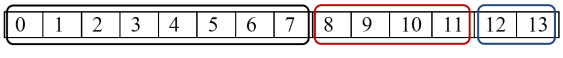
have the same length as $(6,0)$. The only *CodeNumber* whose length is different is 11. So $(6,0)$ pair has ES of 5 in this example.

In CAVLC, *escape mode* is rarely used. While in C2DVLC, it is very frequently used and it may be difficult to find a block in which all the transform coefficients are coded using *regular mode*. For *regular mode* of C2DVLC, ES ranges from 1 to 25 and ES is up to 2^n for *escape mode*, where n is the number of bits in the suffix of Exp-Golomb codeword, while respecting the second constraint.

5.2 Real-time SE-C2DVLC

The scheme of mapping encryptable suffixes to indices works fine if codewords are encrypted separately. But when we make a plaintext by encryptable bits of codewords, the encrypted codewords may not be valid codewords because of the second constraint that C2DVLC encryptable bits do not use a full code-space. For example, for a codeword with $ES = 20$, five bits will be used for its index, with only first 20 valid values $\{0,1,2,...,19\}$ and values $\{20,21,...,31\}$ will not be valid. The encrypted index must lie in the valid range *i.e.*, $\{0,1,2,...,19\}$.

We need a scheme to create a plaintext for real-time framework as shown in Fig. 8. In this section, we propose two schemes to overcome this limitation for real-time SE of C2DVLC codewords in Section 5.2.1 and Section 5.2.2. Both of these algorithms are a compromise between the processing power and security of real-time selective encryption.



Index	Encryption Space (ES)	Encrypted ES
5	1st code-space (0, 1, …,7)	1st (0, 1, …,7)
9	2nd code-space (8, 9, …,11)	2nd (8, 9, …,11)
13	3rd code-space (12,13)	3rd (12,13)

Figure 9: Example of second real-time approach for SE-C2DVLC.

5.2.1 First Approach

First approach targets at utilizing all the available encryption space, at the cost of increased processing power. In this case, we keep track of the available valid ES for indices of codewords in the plaintext X_i. After encryption, we have two types of encrypted indices: valid and non-valid. We replace the valid encrypted indices by the new ones, while the non-valid encrypted indices remain there in the plaintext and are encrypted again to get the valid encrypted indices.

For example, for a 5-bit index with ES = 20 having valid values $\{0,1,2,...19\}$, if the encrypted index lies in first 20 values, it is valid. Otherwise we will encrypt it again till the encrypted index lies in valid range.

This scheme works fine because we know the valid and non-valid values for each index on the decoder side too. If it takes multiple iterations to get a valid encrypted index on the encoder side, it will take exactly the same number of iterations on the decoder side to get the original index, since all the index values between them will not be valid. On the decoder side, we use the same indication of being 'valid' to stop the decryption iterations for a particular index.

5.2.2 Second Approach

Second approach aims at saving the processing power, at the cost of a smaller ES. In this approach, if ES is not full (not power of 2), it is decomposed into smaller full code-spaces. Let us suppose, L is the non-full code-space which is to be decomposed to several smaller full code-spaces $l_{full}[n]$. The code-space $l_{full}[i]$ that contains the index value to be encrypted, is the available encryption space in this approach. Let 2^k be the highest power of 2 lesser or equal to L (k is such that $2^k \leq L$), it makes the first encryption space $l_{full}[1]$. This process is repeated on the remaining ES ($L-2^k$) recursively to decompose L into full code-spaces (which are power of 2).

For example, a non-full code-space with ES = 14 will be decomposed into three full code-spaces having encryption spaces 8, 4 and 2 respectively as shown in Fig. 9. If the index value is 5, its ES will be the first full code-space $\{0,1,2,...7\}$ with $ES = 8$ and its encrypted index will also lie in the same full code-space. Similarly if the index value is 9 or 13, their ESs will be $\{8,9,..,11\}$ and $\{12,13\}$ with $ES = 4$ and $ES = 2$ respectively. Their encrypted index values will also lie in the respective code-spaces as explained in Fig. 9. It is important to note that we can have the final full code-space to be $ES = 1$. In that case, the final index value in the ES will not be encrypted. For example, for ES = 15, the code-spaces will be of sizes 8, 4, 2 and 1. In this case, the final index value cannot be encrypted because it lies in code-space with $ES = 1$.

After encryption with AES cipher in the CFB mode, original bits in the bitstream are substituted by the corresponding encrypted bits.

5.3 Experimental results

For the experimental results, nine benchmark video sequences have been used for the analysis in QCIF format.We have presented analysid of SE-C2DVLC for *intra* sequence and *intra & inter* sequence in Section 5.3.1 and Section 5.3.2 respectively. It also contains analysis of encryption space (ES) and required processing power for first and second SE approaches.

5.3.1 *Intra* frames

To demonstrate the efficiency of our proposed scheme for *intra* frames, we have compressed 100 frames of each sequence at 30 fps as *intra*. Table 1 compares the PSNR of 100 frames of nine video sequences at QP value of 28 without encryption and with both SE approaches. One can note that the proposed algorithm works well for all types of video sequences having various combinations of motion, texture and objects. Table 1 compares the PSNR of 100 frames of nine video sequences at QP value of 28 without encryption and with both SE approaches. One can note that the proposed algorithm works well for all types of video sequences having various combinations of motion, texture and objects. The average PSNR of all the sequences encrypted by first and second approach is 10.2 dB and 10.3 dB for *luma* respectively. Fig. 10 shows the encrypted video frames at different QP values for *foreman* for second approach. PSNR comparison over whole range of QP values is given in Table 2. One can note that, PSNR of the SE video remains in the same lower range (around 10 dB on average for *luma*) for all QP values.

Table 3 demonstrates the encryption space (percentage of bitstream which is encrypted) for both proposed approaches of SE for *intra* frames. For first approach, average ES is 27.65% while it is 25.19% for second approach. Hence to get full code-space, ES of second approach is reduced by 8.9% $((ES1 - ES2)/ES1)$.

5.3.2 *Intra & inter* frames

For experimental evaluation of *intra & inter* frames, *intra period* is set to 10 in a sequence of 100 frames. Table 4 verifies the performance of our algorithm for all video sequences for *Intra & Inter* frames at QP value of 28. Average PSNR of *luma* for all the encrypted sequences is 10.43 dB. Results shown in Table 5 verify the effectiveness of our scheme over the whole range of QP values for *foreman* for *intra & inter* frames.

Table 7 demonstrates the compromise for ES and processing power between first and second approaches of SE. For first approach, average ES is 12.93% while it is 12.28% for second approach. Hence to get complexity reduction, we have to reduce our ES by 5%. For encoding process, we need 1.01% more processing power, while it is 0.61% for second approach. For decoding process, we need 5.10% and 3.73% more processing power for first and second SE approaches respectively.

6 Summary

In this chapter, we have presented the joint C2DVLC entropy coding and selective encryption for AVS. We start with a brief introduction of AVS and its C2DVLC entropy coding. It is followed by introduction of state of the art AES cipher and its CFB mode. After a detailed discussion on recent work in joint entropy coding and selective encryption, we propose an algorithm for joint entropy coding and selective encryption of C2DVLC of AVS. The proposed methods have 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 security in each frame, while maintaining the full bitstream compliance, under a minimal set of computational requirements. For AVS, the proposed encryption algorithm is format complaint and keeps the bitrate unchanged (Shahid et al., 2010b),

Table 1: Comparison of PSNR without encryption with first and second real-time SE-C2DVLC approaches for benchmark video sequences at QP = 28 for *intra*.

	PSNR (Y) (dB)			PSNR (U) (dB)			PSNR (V) (dB)		
Seq.	Orig.	1^{st}	2^{nd}	Orig.	1^{st}	2^{nd}	Orig.	1^{st}	2^{nd}
Bus	37.9	7.8	8.5	41.6	26.0	26.4	42.8	27.9	27.9
City	38.1	12.3	12.6	42.9	30.7	30.7	44.2	31.1	31.0
Crew	39.5	10.2	10.3	41.8	25.0	25.2	40.8	22.2	22.4
Football	39.1	11.9	12.0	41.5	16.3	16.1	42.3	24.1	23.8
Foreman	38.9	9.1	8.8	42.1	23.8	24.1	43.9	26.2	26.9
Harbour	37.8	9.8	9.9	42.2	24.4	25.1	43.6	32.5	31.8
Ice	41.4	10.7	10.8	44.5	26.2	25.8	44.8	20.3	19.1
Mobile	37.9	8.7	8.8	38.6	14.5	14.5	38.4	11.8	12.1
Soccer	38.3	11.4	11.3	42.9	22.1	20.8	44.3	24.1	24.4
Avg.	38.8	10.2	10.3	42.0	23.2	23.2	42.8	24.5	24.4

Table 2: Comparison of PSNR without encryption and with first and second real-time SE-C2DVLC approaches for *foreman* at different QP values for *intra* .

	PSNR (Y) (dB)			PSNR (U) (dB)			PSNR (V) (dB)		
QP	Orig.	1^{st}	2^{nd}	Orig.	1^{st}	2^{nd}	Orig.	1^{st}	2^{nd}
12	49.6	9.0	8.8	50.1	24.8	24.4	50.8	21.5	21.1
20	44.1	8.9	8.7	45.7	26.3	25.9	47.4	22.1	22.8
28	38.9	9.1	8.8	42.1	23.8	24.1	43.9	26.2	26.9
36	34.4	8.9	9.0	39.3	23.8	23.9	40.2	22.0	21.5
44	30.6	9.1	9.7	37.1	23.9	23.9	37.3	21.7	21.3
52	27.0	10.0	9.8	35.3	25.5	25.1	35.9	20.8	20.1

Table 3: Analysis of ES and required processing power with first and second real-time SE-C2DVLC approaches.

Seq.	Encryption Space	
	1^{st} approach (%)	2^{nd} approach (%)
Bus	31.89	28.64
City	27.19	25.46
Crew	20.80	19.80
Football	26.88	24.47
Foreman	24.89	22.91
Harbour	32.10	28.75
Ice	27.27	24.78
Mobile	33.20	28.86
Soccer	24.65	23.02
Avg.	27.65	25.19

Table 4: Comparison of PSNR without encryption with first and second real-time SE-C2DVLC approaches for benchmark video sequences at QP = 28 for *intra* & *inter* with intra period = 10.

	PSNR (Y) (dB)			PSNR (U) (dB)			PSNR (V) (dB)		
Seq.	Orig.	1^{st}	2^{nd}	Orig.	1^{st}	2^{nd}	Orig.	1^{st}	2^{nd}
Bus	36.5	8.0	7.0	41.8	25.2	26.0	43.1	28.0	27.4
City	36.9	12.1	12.3	43.2	31.1	30.4	44.4	31.7	30.9
Crew	38.3	13.4	10.4	42.0	25.4	25.4	40.9	22.4	23.5
Football	37.9	11.8	12.8	41.5	15.2	16.9	42.4	23.4	23.8
Foreman	37.9	8.6	8.2	42.4	25.0	24.4	44.2	26.1	27.2
Harbour	36.2	9.8	9.9	42.4	25.0	28.0	43.9	31.4	33.3
Ice	40.2	10.3	10.8	44.7	26.4	26.1	45.0	18.8	19.8
Mobile	36.1	8.5	9.1	38.8	14.8	12.8	38.5	12.3	11.8
Soccer	37.2	11.5	10.5	43.1	20.4	19.9	44.5	24.2	25.5
Avg.	37.5	10.4	10.1	42.2	23.2	23.3	43.0	24.2	24.8

Table 5: Comparison of PSNR without encryption with first and second real-time SE-C2DVLC approaches for *foreman* at different QP values for *intra* & *inter* with intra period = 10.

	PSNR (Y) (dB)			PSNR (U) (dB)			PSNR (V) (dB)		
QP	Orig.	1^{st}	2^{nd}	Orig.	1^{st}	2^{nd}	Orig.	1^{st}	2^{nd}
12	47.2	9.3	8.7	50.0	25.0	24.7	50.5	23.5	21.2
20	42.8	8.9	8.3	46.0	26.4	27.5	47.7	20.6	23.1
28	37.9	8.6	8.2	42.4	24.9	24.4	44.2	26.1	27.2
36	34.0	8.1	8.7	39.5	23.9	24.9	40.5	21.6	22.3
44	30.4	9.8	8.2	37.3	25.4	23.3	37.7	20.1	23.5
52	27.0	10.7	9.1	35.7	24.4	25.0	36.0	19.8	22.2

Table 6: Analysis of ES and required processing power with first and second real-time SE-C2DVLC approaches.

	Encryption Space I+P		Processing Power (I+P) encoder		decoder	
Seq.	1^{st} (%)	2^{nd} (%)	1^{st} (%)	2^{nd} (%)	1^{st} (%)	2^{nd} (%)
Bus	11.93	11.22	1.38	0.80	5.66	4.04
City	13.38	12.93	1.00	0.62	5.04	3.67
Crew	12.58	12.33	0.62	0.41	3.78	2.66
Football	16.06	14.94	0.82	0.43	5.19	3.91
Foreman	13.61	13.01	0.94	0.57	4.79	3.59
Harbour	12.38	11.72	1.10	0.66	5.48	3.90
Ice	13.07	12.36	0.82	0.46	4.74	3.55
Mobile	11.12	10.28	1.52	1.01	6.50	4.84
Soccer	12.22	11.76	0.88	0.53	4.76	3.44
Avg.	12.93	12.28	1.01	0.61	5.10	3.73

Table 7: Analysis of ES and required processing power with first and second real-time SE-C2DVLC approaches.

Seq.	Encryption Space				Processing Power (I+P)			
	I		I+P		encoder		decoder	
	1^{st} (%)	2^{nd} (%)	1^{st} (%)	2^{nd} (%)	1^{st} (%)	2^{nd} (%)	1^{st} (%)	2^{nd} (%)
Bus	31.89	28.64	11.93	11.22	1.38	0.80	5.66	4.04
City	27.19	25.46	13.38	12.93	1.00	0.62	5.04	3.67
Crew	20.80	19.80	12.58	12.33	0.62	0.41	3.78	2.66
Football	26.88	24.47	16.06	14.94	0.82	0.43	5.19	3.91
Foreman	24.89	22.91	13.61	13.01	0.94	0.57	4.79	3.59
Harbour	32.10	28.75	12.38	11.72	1.10	0.66	5.48	3.90
Ice	27.27	24.78	13.07	12.36	0.82	0.46	4.74	3.55
Mobile	33.20	28.86	11.12	10.28	1.52	1.01	6.50	4.84
Soccer	24.65	23.02	12.22	11.76	0.88	0.53	4.76	3.44
Avg.	27.65	25.19	12.93	12.28	1.01	0.61	5.10	3.73

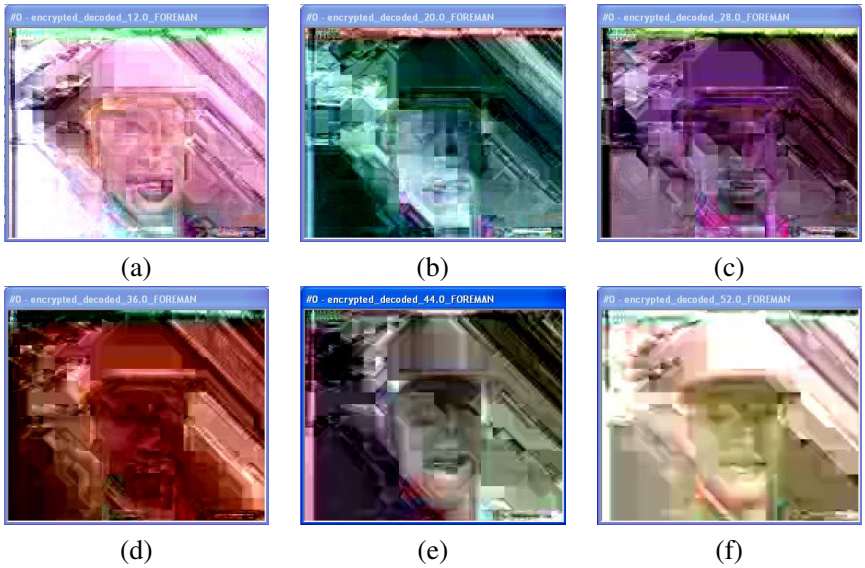


Figure 10: Encrypted video using second real-time SE-C2DVLC approach: *foreman* frame # 0 with QP value: (a) 12, (b) 20, (c) 28, (d) 36, (e) 44, (f) 52.

but creation of plaintext from encryptable bits for real-time implementation is not straight forward, because code-space of C2DVLC is not full. To cope with this limitation, two real-time schemes have been proposed for its selective encryption ,which provides a trade-off between encryption space and processing power (Shahid et al., 2010a).

References

Abomhara, M., Zakaria, O., Khalifa, O., Zaiden, A., & Zaiden, B. (2010). Enhancing Selective Encryption for H.264/AVC Using Advanced Encryption Standard. *International Journal of Computer and Electrical Engineering*, 2(2), 223–229.

Carrillo, P., Kalva, H., & Magliveras, S. (2009). Compression Independent Reversible Encryption for Privacy in Video Surveillance. *EURASIP Journal on Information Security*, 2009, 13.

Cheng, H. & Li, X. (2000). Partial Encryption of Compressed Images and Videos. *IEEE Transactions on Signal Processing*, 48(8), 2439–2445.

Daemen, J. & Rijmen, V. (2002). *AES Proposal: The Rijndael Block Cipher*. Technical report, Proton World Int.l, Katholieke Universiteit Leuven, ESAT-COSIC, Belgium.

Droogenbroeck, M. & Benedett, R. (2002). Techniques for a Selective Encryption of Uncompressed and Compressed Images. In *Proc. Advanced Concepts for Intelligent Vision Systems* (pp. 90–97). Ghent, Belgium.

Dufaux, F. & Ebrahimi, T. (2008). Scrambling for privacy protection in video surveillance systems. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(8), 1168 –1174.

Fan, L., Ma, S., & Wu, F. (2004). Overview of AVS Video Standard. In *Proc. IEEE International Conference on Multimedia and Expo* (pp. 423–426). Taipei, Taiwan.

Fisch, M., Stgner, H., & Uhl, A. (2004). Layered Encryption Techniques for DCT-Coded Visual Data. In *Proc.* 12^{th} *European Signal Processing Conference* (pp. 821–824). Vienna, Austria.

Grangetto, M., Magli, E., & Olmo, G. (2006). Multimedia Selective Encryption by Means of Randomized Arithmetic Coding. *IEEE Transactions on Multimedia*, 8(5), 905–917.

H264 (2003). *Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 ISO/IEC 14496-10 AVC).* Technical report, Joint Video Team (JVT), Doc. JVT-G050.

Jakimoski, G. & Subbalakshmi, K. (2008). Cryptanalysis of Some Multimedia Encryption Schemes. *IEEE Transactions on Multimedia*, 10(3), 330–338.

Jiangtao, W., Hyungjin, K., & Villasenor, J. (2006). Binary Arithmetic Coding with Key-based Interval Splitting. *IEEE Signal Processing Letters*, 13(2), 69–72.

Li, C., Zhou, X., & Zong, Y. (2008). NAL Level Encryption for Scalable Video Coding. *Lecture notes in Computer Science, Springer*, (5353), 496–505.

Li, W. (2001). Overview of Fine Granularity Scalability in MPEG-4 Video Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(3), 301–317.

Lian, S., Liu, Z., Ren, Z., & Wang, H. (2007). Commutative Encryption and Watermarking in Video Compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(6), 774–778.

Lian, S., Liu, Z., Ren, Z., & Wang, Z. (2005). Selective Video Encryption Based on Advanced Video Coding. *Lecture notes in Computer Science, Springer-verlag*, (3768), 281–290.

Lin, E., Eskicioglu, A., Lagendijk, R., & Delp, E. (2005). Advances in Digital Video Content Protection. *Proc. of the IEEE*, 93(1), 171–183.

Lookabaugh, T. & Sicker, D. (2004). Selective Encryption for Consumer Applications. *IEEE Communications Magazine*, 42(5), 124–129.

Lukac, R. & Plataniotis, K. (2005). Bit-Level Based Secret Sharing for Image Encryption. *Pattern Recognition*, 38(5), 767–772.

Ma, S. & Gao, W. (2006). Low Complexity Integer Transform and Adaptive Quantization Optimization. *Journal of Computer Science and Technology*, 21(3), 354–359.

Maniccam, S. & Bourbakis, N. (2004). Image and Video Encryption using SCAN Patterns. *Pattern Recognition*, 37(4), 725 –737. Agent Based Computer Vision.

Martin, K., Lukac, R., & Plataniotis, K. (2005). Efficient Encryption of Wavelet-Based Coded Color Images. *Pattern Recognition*, 38(7), 1111–1115.

Ou, S., Chung, H., & Sung, W. (2006). Improving the Compression and Encryption of Images using FPGA-Based Cryptosystems. *Multimedia Tools and Applications*, 28(1), 5–22.

Rodrigues, J., Puech, W., & Bors, A. (2006). Selective Encryption of Human Skin in JPEG Images. In *Proc. IEEE International Conference on Image Processing* (pp. 1981–1984). Atlanta, USA.

Said, A. (2005). Measuring the Strength of Partial Encryption Scheme. In *Proc. IEEE International Conference on Image Processing*, volume 2 (pp. 1126–1129). Genova, Italy.

Shahid, Z., Chaumont, M., & Puech, W. (2010a). Over the Real-Time Selective Encryption of AVS Video Coding Standard. In *Proc. European Signal Processing Conference* Aalborg, Denmark.

Shahid, Z., Chaumont, M., & Puech, W. (2010b). Selective Encryption of C2DVLC of AVS Video Coding Standard for I & P Frames. In *Proc. IEEE International Conference on Multimedia & Expo* (pp. 1655–1660). Singapore.

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

Tang, L. (1996). Methods for Encrypting and Decrypting MPEG Video Data Efficiently. In *Proc. ACM Multimedia*, volume 3 (pp. 219–229). New York, NY, USA.

Wang, Q., Zhao, D., & Gao, W. (2006). Context-based 2D-VLC Entropy Coder in AVS Video Coding Standard. *Journal of Computer Science and Technology*, 21(3), 315–322.

Wang, Q., Zhao, D., Ma, S., Lu, Y., Huang, Q., & Ga, W. (2004). Context-based 2D-VLC for Video Coding. In *Proc. IEEE International Conference on Multimedia and Expo* (pp. 89–92).

Wang, X., Zheng, N., & Tian, L. (2010). Hash Key-Based Video Encryption Scheme for H.264/AVC. *Signal Processing: Image Communication*, 25(6).

Wang, Y., Wenger, S., Wen, J., & Katsaggelos, A. (2000). Error Resilient Video Coding Techniques. *IEEE Signal Processing Magazine*, 17(4), 61 –82.

Wen, J., Severa, M., Zeng, W., Luttrell, M., & Jin, W. (2002). A Format-Compliant Configurable Encryption Framework for Access Control of Video. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(6), 545–557.

Wu, G., Wang, Y., & Hsu, W. (2005). Robust Watermark Embedding/Detection Algorithm for H.264 Video. *Journal of Electronic Imaging*, 14(1).

Wu, M., Joyce, R., & Kung, S. (2000). Dynamic Resource Allocation via Video Content and Short-Term Traffic Statistics. In *Proc. IEEE International Conference on Image Processing*, volume 3 (pp. 58–61).

Yabuta, K., Kitazawa, H., & Tanaka, T. (2005). A New Concept of Security Camera Monitoring with Privacy Protection by Masking Moving Objects. In *Proc. Advances in Multimedia Information Processing*, volume 1 (pp. 831–842).

Yeo, B. & Liu, B. (1995). Rapid scene analysis on compressed video. *IEEE Transactions on Circuits and Systems for Video Technology*, 5(6), 533 –544.

Zeng, W. & Lei, S. (2003). Efficient Frequency Domain Selective Scrambling of Digital Video. *IEEE Transactions on Multimedia*, 5, 118–129.

Zhang, L., Wang, Q., Zhang, N., Zhao, D., Wu, X., & Gao, W. (2009). Context-based Entropy Coding in AVS Video Coding Standard. *Image Communication*, 24(4), 263–276.