OVER THE REAL-TIME SELECTIVE ENCRYPTION OF AVS VIDEO CODING STANDARD

Z. Shahid, M. Chaumont and W. Puech

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

ABSTRACT

Audio Video Coding Standard (AVS) is the emerging video standard in which 2-dimensional context adaptive variable length coding (C2DVLC) is used. The main difference between context adaptive variable length coding (CAVLC) of H.264/AVC and C2DVLC of AVS from 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 can be assigned indices and are encrypted. This scheme 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 C2DVLC encryptable bits do not use a full code-space. In this paper, we will present two schemes to overcome this limitation for real-time implementation of C2DVLC codewords. Both of the algorithms are a compromise between the processing power and security of realtime selective encryption.

1. 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. Encryption is used to restrict access of digital data to authenticated users only. For video data, the concept of selective encryption (SE) has evolved in which only a small part of the whole bitstream is encrypted [11]. In this work, we have transformed C2DVLC module of AVS into crypto-compression module by the encryption of non-zero coefficients (NZs).

SE of state of the art video codec like H.264/AVC has been discussed in literature. Lian et *al.* have done partial encryption of some fields of H.264/AVC as intra-prediction mode, residue data, inter-prediction mode and motion vectors [6]. Carrillo et *al.* [3] have also presented an idea of encryption for H.264/AVC. They do permutations of the pixels of those macro-blocks (MBs) which are in ROI. The drawback of this scheme is that the 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.

The use of general entropy coder as an encryption step has been studied in [15]. It encrypts NZs by using different Huffman tables for each input symbols. The tables, as well as the order in which they are used, are kept secret. This technique is vulnerable to known plaintext attack as explained in [5]. For H.264/AVC, entropy coding based SE has been discussed for context adaptive variable length coding (CAVLC) [9] and context adaptive binary arithmetic coding (CABAC) [10] which fulfills real-time constraints by producing format-complaint encrypted bitstream without changing the bitrate. In Section 2, overview of AVS and C2DVLC is presented. Comparison of AVS with H.264/AVC is also presented in this section. We explain the proposed algorithms for realtime SE of AVS in Section 3. Section 4 contains the experimental evaluation, followed by the concluding remarks in Section 5.

2. PRELIMINARIES

2.1 Overview of AVS video coding standard

AVS [4] is the state of the art video coding standard of China and is based on motion compensated hybrid framework. It has slightly lesser performance but is less complex than H.264/AVC [2] video standard of ITU-T and ISO/IEC. A video frame is processed into blocks of 16x16 pixels, called macroblock (MB). Each MB can be encoded as *intra* or *inter*.

In *intra* frame, spatial prediction is performed from reconstructed (i.e. top and left) MBs. It is performed on blocks of 8x8, in contrast to 4x4 and 16x16 block size in H.264/AVC. Spatial prediction in AVS is less complex with only five modes for *luma*, as compared to thirteen modes in H.264/AVC. Reference pixels, which are to be used for prediction, are first low pass filtered. In *inter* mode, motion compensated prediction is done from previous frames. It supports variable block size motion estimation up to 8x8 block, quarter pixel motion estimation and multiple reference frames in *inter* frame.

The difference between original and predicted frame is called a *residual*. This *residual* is coded using transform coding. In AVS, standard DCT transform has been replaced by 8x8 Integer Cosine Transform (ICT) [7] which does not need any multiplication operation and can be implemented by only additions and shifts. It is followed by quantization and zigzag scan. For quantization, QP value ranges 0-63 with a period of approximate 8. In AVS Part-2, two modes for entropy coding are supported, namely C2DVLC in Jizhun profile and context-based binary arithmetic coding (CBAC) in Jiaqiang profile [16]. In the last step, either of the entropy coding techniques namely C2DVLC or CBAC is used.

On the decoding side, compressed bitstream is decoded by entropy decoding module, followed by inverse-zigzag scan. These coefficients are then inverse-quantized and inverse transformed to get the *residual* signal which is added to the predicted signal to reconstruct the original signal back. AVS decoder complexity is further reduced by moving the inverse scaling from decoder to encoder module.

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

 ISSN 2076-1465

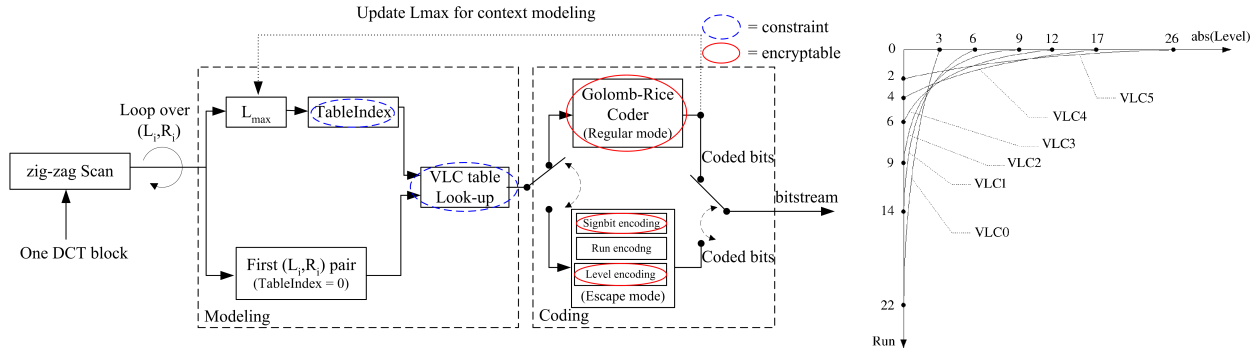


Figure 1: a) Block diagram of C2DVLC showing constraints and encryptable blocks, b) Limit of each 2D-VLC table of C2DVLC.

sequences [14]. 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 distinguish it from H.264/AVC.

2.2 Context-based 2D Variable Length Coding

In this standard, an efficient context-based 2D-VLC entropy coder is designed for coding 8x8 block-size transform coefficients, where 2D-VLC means that a pair of Run-Level (L_i, R_i) is regarded as one event and jointly coded [13].

Fig. 1.a shows the working of C2DVLC by a flow graph. A 2-D DCT block is converted to 1-D array by zig-zag scan. It is followed by Run-Length coding which transforms this array to (L_i, R_i) pairs. C2DVLC starts 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. 1.b. 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 look-up table 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, $(L_i, R_i) = (2, 1)$ is mapped to *CodeNumber* 11 and $(L_i, R_i) = (-2, 1)$ is mapped to *CodeNumber* is 12. This *CodeNumber* is then coded by Exp-Golomb code. 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:

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

with the threshold for each table given as:

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

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.

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 a 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 1 kilo bytes. This method gives gain up to 0.23 dB compared to one-table-for-one-type-of-block coding method [12]. For further details about C2DVLC, please refer to [16].

2.3 C2DVLC vs. CAVLC

The common point between C2DVLC of AVS and CAVLC of H.264/AVC is that both of them are adaptive to the local statistics of DCT coefficients and coding efficiency of both of them is similar. Otherwise C2DVLC is substantially different as compared to CAVLC. In CAVLC, Exp-Golomb coding is used only for coding of syntax elements only. Transform coefficients are converted to levels and runs, which are coded **separately** using multiple VLC tables.

While in AVS, Exp-Golomb coding is used for coding all syntax elements including transform coefficients. 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. THE PROPOSED ALGORITHM

SE of C2DVLC codewords is being presented in Section 3.1, while the proposed schemes for its real-time implementation are discussed in Section 3.2.

3.1 Selective Encryption of C2DVLC

To keep the bitrate unchanged alongwith the 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. 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. In this mode, code-space is not full because of two major limitations. First, we do not have specific bits to be encrypted and the encryption space (ES) is not a power of 2. Second, non-consecutive *CodeNumbers* are assigned to consecutive levels. 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. Fig. 1.a encircles the functional blocks with constraints. It also shows the encryptable functional blocks.

Advanced Encryption Standard (AES) algorithm is used in Cipher Feedback (CFB) mode for encryption. 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). The keystream element Z_i is then generated and the ciphertext block Y_i is produced as:

$$\begin{cases} Z_i &= E_k(Y_{i-1}), for\ i \geq 1 \\ Y_i &= X_i \oplus Z_i \end{cases}, \qquad (3)$$

where $\oplus$ is the XOR operator.

For example, let us code current pair $(L_i, R_i) = (6, 0)$ with $TableIndex = 3$ for *intra_luma* mode as shown in Fig. 2. 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* 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.

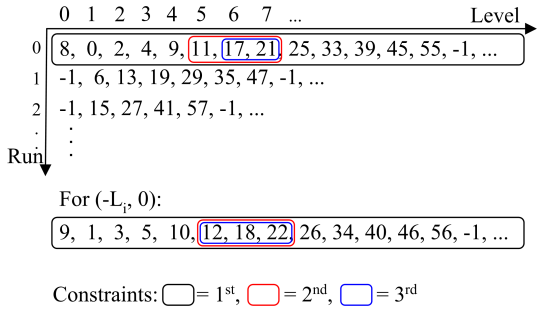


Figure 2: (L_i, R_i) encryption in *regular mode* for *Table* with $TableIndex = 3$.

3.2 Real-time Selective Encryption of C2DVLC

For real-time implementation, we create the plaintext X_i first by putting the indices of codewords from C2DVLC bitstream, which is then encrypted to create Y_i and then substitution of the original codewords with the encrypted information is performed. In state of the art video coding standards like AVS, spatial prediction is performed for *intra* frames. We can encode a single video frame into several independent coding units called slices and prediction is performed within a slice only. Since slice is an independent coding unit, SE is performed on them independently. Let C, the length of the plaintext X_i, is 128 and $L(X_i)$ is 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 $j \in \{L(X_i) + 1, \ldots, C\}$, to fill in the vector X_i with zeros up to C bits.

In case of C2DVLC: 1) ES is not contiguous i.e., we do not have consecutive *CodeNumbers* for consecutive transform coefficient values. This limitation is overcome by using indices instead of *CodeNumbers*. 2) ES is not always full (not equal to 2^n). 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.

The situation becomes worse when we make the plaintext from these indices. So for every index, we will be having few valid values, while others will not be valid and we cannot accept them as encrypted index values. In this case, we can have one of the two approaches presented in Section 3.2.1 and Section 3.2.2.

3.2.1 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.

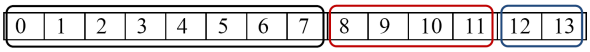
This scheme works fine because we know the valid and non-valid values for each index on the decoder side too. If it takes multiple iterations to get a valid encrypted index on the encoder side, it will take exactly the same number of itera-

tions on the decoder side to get the original index, since all the index values inbetween 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.

3.2.2 *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 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. 3. 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 $\{8,9,..,11\}$ & $\{12,13\}$ with $ES = 4$ & $ES = 2$ respectively. Their encrypted index values will also lie in the respective code-spaces as explained in Fig. 3. 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 $ES = 1$.



ES = 14, Full code-spaces = 1st, = 2nd, = 3rd

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 3: Example of 2^{nd} approach of realtime SE.

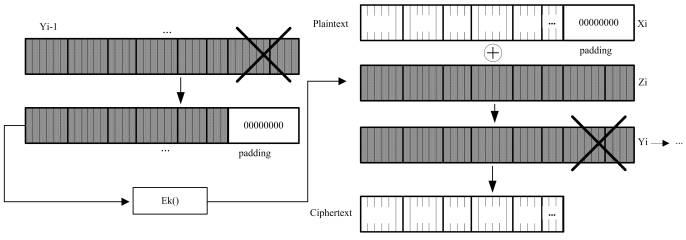


Figure 4: Real-time SE of C2DVLC of AVS.

After encryption with AES cipher in the CFB mode as illustrated in Fig. 4, original bits in the bitstream are substituted by the corresponding encrypted bits.

4. EXPERIMENTAL RESULTS

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, color, contrast and objects. We have used the AVS version RM 6.2c [1] and have performed SE of C2DVLC for *intra & inter* frames. It also contains the ES and processing power compromise for first and second SE approaches.

4.1 *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 type 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*. Fig. 5 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 5 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 $8.9\%((ES1-ES2)/ES1)$.

4.2 *Intra & Inter* Frames

For experimental evaluation of *intra & inter* frames, *intra period* is set to 10 in a sequence of 100 frames. Table 3 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 4 verify the effectiveness of our scheme over the whole range of QP values for *foreman* for *intra & inter* frames.

Table 5 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 2nd Approach. For decoding process, we need 5.10% and 3.73% more processing power for first and second SE approaches respectively.

5. CONCLUSION

In this paper, a realtime framework for SE of AVS based on C2DVLC has been presented. Since code-space of C2DVLC is not full, two schemes have been proposed for its real-time encryption. First approach utilizes all the available ES, while utilizing 0.5% more processing power as compared to 2nd approach, while ES of 2nd approach gets reduced by 5% for (I+P) frames. Second approach is recommended for handheld devices because of lesser requirement of processing power. Since all the constraints posed by the contexts and Exp-Golomb codewords for each NZ, have been fulfilled, encrypted bitstream is fully compliant to AVS format and is

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

Seq.	PSNR (Y) (dB)			PSNR (U) (dB)			PSNR (V) (dB)		
	Orig.	1st	2nd	Orig.	1st	2nd	Orig.	1st	2nd
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 SE approaches for *foreman* at different QP values for *intra* .

QP	PSNR (Y) (dB)			PSNR (U) (dB)			PSNR (V) (dB)		
	Orig.	1st	2nd	Orig.	1st	2nd	Orig.	1st	2nd
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: Comparison of PSNR without encryption and with first and second SE approaches for benchmark video sequences at QP = 28 for *intra* & *inter* with intra period = 10.

Seq.	PSNR (Y) (dB)			PSNR (U) (dB)			PSNR (V) (dB)		
	Orig.	1st	2nd	Orig.	1st	2nd	Orig.	1st	2nd
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 4: Comparison of PSNR without encryption and with first and second SE approaches for *foreman* at different QP values for *intra* & *inter* with intra period = 10.

QP	PSNR (Y) (dB)			PSNR (U) (dB)			PSNR (V) (dB)		
	Orig.	1st	2nd	Orig.	1st	2nd	Orig.	1st	2nd
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 5: Analysis of ES and required processing power for first and second SE approaches.

Seq.	Encryption Space				Processing Power (I+P)			
	I		I+P		encoder		decoder	
	1st (%)	2nd (%)	1st (%)	2nd (%)	1st (%)	2nd (%)	1st (%)	2nd (%)
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

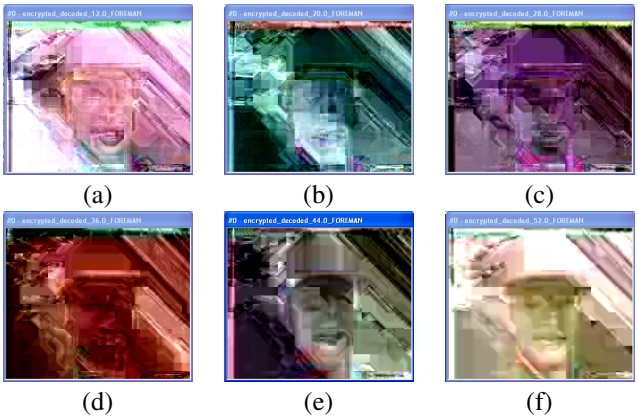


Figure 5: Encrypted video using second approach: *foreman* frame # 0 with QP value: a) 12, b) 20, c) 28, d) 36, e) 44, f) 52.

decodable by reference AVS decoder. Realtime constraints have been successfully fulfilled by having exactly the same bitrate. The experiments have shown that we can achieve the desired level of encryption in each frame, while maintaining the full AVS video coding standard compliance for both of the proposed approaches. The proposed schemes can be extended to protect only ROI in video surveillance and can be applied to medical image transmission [8].

Acknowledgment

This work is in part supported by the VOODDO (2008-2011) project of the french Agence Nationale pour la Recherche.

REFERENCES

[1] http://159.22.42.57/incoming/dropbox/video_software/Rm62c.zip.

[2] 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, March 2003.

[3] P. Carrillo, H. Kalva, and S. Magliveras. Compression Independent Object Encryption for Ensuring Privacy in Video Surveillance. In *ICME*, 2008.

[4] L. Fan, S. Ma, and F. Wu. Overview of AVS Video Standard. In *Proc. IEEE International Conference on Multimedia and Expo*, pages 423–426, Taipei, Taiwan, 2004.

[5] G. Jakimoski and K. Subbalakshmi. Cryptanalysis of Some Multimedia Encryption Schemes. *IEEE Transactions on Multimedia*, 10(3):330–338, April 2008.

[6] S. L. S. Z. Liu, Z. Ren, and Z. Wang. Selective Video Encryption Based on Advanced Video Coding. *Lecture notes in Computer Science, Springer-verlag*, (3768):281–290, 2005.

[7] S. Ma and W. Gao. Low Complexity Integer Transform and Adaptive Quantization Optimization. *J. Comput. Sci. Technol.*, 21(3):354–359, 2006.

[8] W. Puech and J. Rodrigues. A New Crypto-Watermarking Method for Medical Images Safe Transfer. In *Proc. 12th European Signal Processing Conference (EUSIPCO'04)*, pages 1481–1484, Vienna, Austria, 2004.

[9] 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.

[10] Z. Shahid, M. Chaumont, and W. Puech. Fast Protection of H.264/AVC by Selective Encryption of CABAC for I & P frames. In *Proc. 17th European Signal Processing Conference (EUSIPCO'09)*, pages 2201–2205, Glascow, Scotland, 2009.

[11] A. Uhl and A. Pommer. *Image and Video Encryption: From Digital Rights Management to Secured Personal Communication*. Springer, 2005.

[12] Q. Wang, D. Zhao, S. Ma, Y. Lu, Q. Huang, and W. Ga. Context-based 2D-VLC for Video Coding. In *Proc. IEEE International Conference on Multimedia and Expo*, volume 1, pages 89–92 Vol.1, June 2004.

[13] Q. Wang, D.-B. Zhao, and W. Gao. Context-based 2D-VLC Entropy Coder in AVS Video Coding Standard. *J. Comput. Sci. Technol.*, 21(3):315–322, 2006.

[14] X. Wang and D. Zhao. Performance Comparison of AVS and H.264/AVC Video Coding Standards. *J. Comput. Sci. Technol.*, 21(3):310–314, 2006.

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

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