Compliant selective encryption for H.264/AVC video streams

Cyril Bergeron and Catherine Lamy-Bergot
EDS/SPM (Signal Processing and Multimedia department)
THALES Communications France
Colombes, France
{cyril.bergeron, catherine.lamy}@fr.thalesgroup.com

Abstract—**This paper presents a novel selective encryption scheme that allows for secure transmission of video streams such as H.264/AVC ones, while keeping complete backward compatibility with the corresponding video standard. This compatibility allows consequently any standard compliant decoder to decode the ciphered stream without specific provision in terms of resynchronization procedure, albeit incorrectly from the visual point of view. The bits to be encrypted are chosen with respect to the video standard, the full compatibility being achieved by encrypting bits for which the impact of each of the encrypted configurations gives a non-desynchronized and fully standard compliant bitstream.**

Numerical results obtained with this partial ciphering algorithm in the context of H.264/AVC show that PSNR degradations of about 25 to 30 dB are obtained when the ciphering key is unknown.

Keywords— Security, ciphering, videociphering, H.264/AVC

I. INTRODUCTION

The exchanges of multimedia data are currently growing at a previously unknown speed, leading to an increasing demand for remote video communication and to the development of systems aiming at providing confidential and reliable information exchanges. The security aspects implied by this need of confidential exchanges are nowadays insufficiently met by the video coding standards which were not built with encryption capabilities. The very nature of video encoding schemes such as MPEG ones, which rely on predictive coding make them difficult candidates [1] for encryption.

Based on the observation that encrypting the whole compressed bitstream is very expensive both in delay and processing time, it is proposed here to only partially encrypt the compressed bitstream, following the path of others [2]. As a matter of fact, although a large portion of the compressed data is left unencrypted, an adequate choice of bits to encrypt still makes it sufficiently difficult to recover the original data without deciphering the encrypted part that the security of the transmission is achieved.

The present algorithm proposes a selective encryption scheme allowing secure transmission of video streams such as H.264/AVC ones while keeping complete compatibility with the video standard, in the sense that any standard compliant codec will fully decode the encrypted bitstream without specific provision in terms of resynchronization or behavior in erroneous environment procedures. Being compatible with the video standard, the encryption scheme will then easily be used in all coding approaches, from already deployed solutions to future ones. Using the proposed method allows to insert the encryption mechanism inside the video encoder, providing a secure transmission which does not alter the transmission process. The bits 'selected for encryption' are chosen with respect to the considered video standard according to the following rule: each of their encrypted configurations gives a non-desynchronized and fully standard compliant bitstream. This can in particular be done by encrypting only parts of the bitstream which have no or a negligible impact (meaning do not lead to desynchronization during the decoding process, and do not imply a major change in decoding contexts) in the evolution of the decoding process, and whose impact is consequently purely a visual one.

This paper is organized as follows. After some considerations on encryption methods for video streams, Section II presents the proposed algorithm, highlighting which bits can be selected for encryption and how the encryption process can be applied to ensure that a full compatibility with the standard is kept. In Section III is given a short presentation of the particular case of H.264/AVC standard, as well as an example of bits 'selected for encryption' is its context. Numerical results for this same standard are then given in Section IV. Finally, Section V draws out some conclusions.

II. ENCRYPTING A VIDEO STREAM

A. *Facing a video stream encryption: existing problems*

A first question that arises when coupling encryption and compression is on the relative place of the two processes. In many cases, the compressed video data is treated as any other data by the encryption mechanism placed after the video encoding process is fully completed, and decrypted at the receiver side before the beginning video decoding process. This method adds latency and involves more computations, as either the whole bitstream is then encrypted, or the bitstream must be segmented into differently processed streams which are then put back together at the decoder side. Consequently, other solutions were introduced that conjugate more intimately the encryption and compression processes. Observing however that encryption solutions applied before the compression mechanism is effectively completed can lead to less efficient compression solutions [6], it is proposed in this article to embed the encryption mechanism inside the compression mechanism, and apply the ciphering after the entropy coding. Then, only the encryption key that has to be transmitted may increase the throughput (typical values [7] should be of less than 0.1%). Another advantage of such a method is that the insertion of the encryption process inside the video (de)coding process allows to take advantage of the then-known syntax meaning of each bit, which hence to decide which parts should be encrypted or not (*resp*. decrypted or not at the decoder side) without having to signal the ciphered parts, as the information used to discriminate parts is the knowledge of the video standard.

B. *Realising the encryption: how to cipher and remain backward compatible*

Encryption or ciphering, result of the cryptography process, aims at keeping messages secure and grant access to the deciphered version only to authorized ones. The original message, or *plaintext*, is transformed into an encrypted message, or *ciphertext*, thanks to an encryption mechanism which generally relies on the use of a *key*, whose secure exchange between the emitter and the receiver guarantees that only receiver can decrypt the ciphertext.

Among the most well known algorithms in cryptography, one finds the block cipher algorithm Advanced Encryption Standard (AES) [4], that must be used with a confidentiality mode such as the counter mode CTR [5] to produce the ciphertext. It relies on eXclusive-ORing (X-ORing) the output block with the plaintext to produce the ciphertext, and vice versa at the decoder. Yet, X-ORing generates ciphertext outputs taking every possible values, as illustrated by Figure 1-(b), which may lead to impossible configurations if the video standard allows only some given configurations as possible, that is to say lead to a non compliant bitstream.

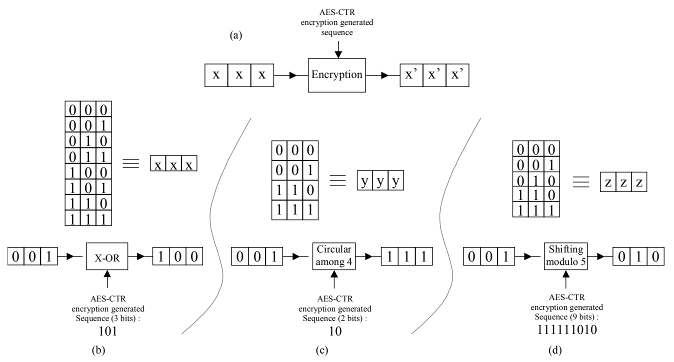
(a)
AES-CTR encryption generated sequence
x x x
Encryption
x' x' x'
000 001 010 011 100 101 110 111
x x x
0 0 1
X-OR
1 0 0
AES-CTR encryption generated Sequence (3 bits) : 101
(b)
000 001 110 111
y y y
0 0 1
Circular among 4
1 1 1
AES-CTR encryption generated Sequence (2 bits) : 10
(c)
000 001 010 110 111
z z z
0 0 1
Shifting modulo 5
0 1 0
AES-CTR encryption generated Sequence (9 bits) : 111111010
(d)

Figure 1 – Encryption process (a) and examples of its application (b)-(d).

A compatible encryption solution will then be to use the output blocks provided by AES CTR not directly to perform the encryption, but to select among the possible configurations (all standard compliant ones) which one should be used as ciphertext for a given plaintext. The possible configurations can be stored in a table, with positions from 0 to n-1. Two cases can then be separated, depending on whether the number of possible configurations is a power of two or not:

1. With n=2^k possible configurations, one sees that k bits can be used to perform encryption. A way to proceed, denoted by *circular*, is to use those *k* bits to select from a position *i* of the plaintext configuration the one at position *i+*2^k *[k]* as ciphertext. AES CTR generating equi-probable output blocks, this circular ciphering still holds good properties from the resistance to cryptanalysis point of view. This scheme is illustrated by Figure 1-(c) for *k=2, i=1*.
2. The problem is more complex when the number of possible configurations is not a power of two, as one can not equally divide 2^k solutions among n when n is not a power of two, which is bad from the resistance to cryptanalysis point of view. A solution is to admit slightly asymmetric repartitions of the permutations. In practice, considering a key of k bits, corresponding to 2^k possible output blocks, and a set of n possible configurations, one can shift from one configuration (i) to another by choosing the 2^k next one modulo n (configuration i+2^k [n]). One configuration is consequently used as ciphertext between $\lceil 2^k/n \rceil$ and $\lfloor 2^k/n \rfloor$ times. This leads to a bias in the probability distribution, defined as the maximal probability difference for the considered distribution with an infinitely random one, that is to say a uniform one, where each configuration has a probability of 1/n. The bias is then given by:
$$\alpha = \max\left(\left| \frac{\left\lfloor \frac{2^k}{n} \right\rfloor + 1}{2^k} - \frac{1}{n} \right|, \left| \frac{\left\lfloor \frac{2^k}{n} \right\rfloor}{2^k} - \frac{1}{n} \right| \right)$$
Figure 1-(d) illustrates this encryption solution for n=5, i=1 and k=9, leading to a bias $\alpha \cong 0.001302$. The value of the accepted bias, hence the value of k to be used can be determined by the wished security level of the application.

It is supposed in the following that efficient encryption mechanisms ensuring a sufficient level of security for the encryption process can be applied based on an exchange of long enough encryption keys and a negligible additional throughput only due to initial key exchanges. Still, for the sake of simplicity, simulations presented in Section IV were made with a random pattern generated with random C function.

C. *Selecting the bits to be ciphered: principle*

The idea is to apply a ciphering that will alter the video stream only visually while keeping it fully decodable even by a non-robust standard compliant decoder, as illustrated by Figure 2. A way to determine which bits can be ciphered could be to test over a set of sequences in which parts in the bitstream bit inversion can be applied without crashing the decoder process, and bringing only visual errors. Obviously this is only a first approach that must be followed by a careful study of the standard and its requirements to determine all or many of the possibilities to encrypt bits without leading to a potentially non-compatible stream.

In practice, the bits to be encrypted are chosen with respect to the considered video standard to ensure full compatibility, achieved by selecting the bits (generally parts of codewords) for which each of the encrypted configuration modify negligibly the decoding process contexts in the sense where their introduction does not create de-synchronization nor lead to non-compliant bitstreams. As such, an encryption operation leading to a change of a symbol table used in the coding process is not-negligible whereas an encryption operation that leads to interpreting a given codeword instead of another of same size is negligible. In each case, it is important to note that the bits shall maintain this capacity in every coded bitstream, and that it can not be envisaged to consider cases where given configuration of bitstream will allow immediate or delayed resynchronization. As such, the bits 'selected for encryption' will correspond to cases were several codewords of same length are available with no major context change when shifting from one to another, and the ciphering will consist to swap on of the bit(s) configuration by another, as illustrated by Figure 1-(a).

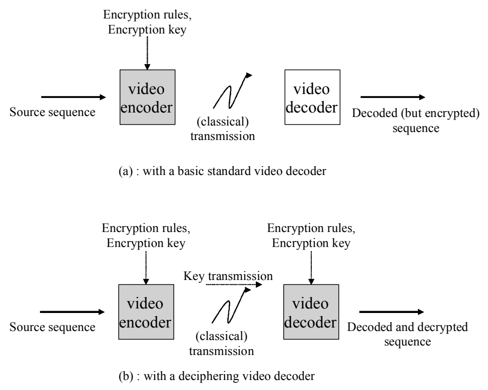

Figure 2 – Principle of coding/transmission/decoding chain with encryption mechanism for decrypting and not-decrypting video decoders.

The interest to choose carefully the way encryption is performed is double: firstly one ensures the compatibility with the requirements of the considered video standard, and secondly one makes it difficult for cryptanalysis attacks to find an angle to break the encryption key, as it is aimed at making all solutions possible, hence removing possibilities to rule out some cases based on non-respect of standard syntax.

III. Application to H.264/AVC standard

A. Presentation of H.264/AVC standard

Common standard [3] established by the MPEG consortium and the Video Coding Expert Group (VCEG), H.264/AVC specifies the video coding aspects of the most efficient video coding tool known as of today. This standard includes two entropy coding modes, the Context-Adaptive Algebraic Coding (CABAC) mode, which relies on algebraic compression and the Context-Adaptive VLC (CAVLC) mode, which relies on more classical variable-length codes. In this article, the CAVLC mode is considered, mode which is the only one considered in all the standard profiles.

Relying basically on the concept of run-length coding well-known from previous video coding standards, the CAVLC mode main originality is that the VLC tables for the various syntax elements are switched depending on the values of previously transmitted syntax elements. Two codes families are used within this mode: Exponential-Golomb (Exp-Golomb) codes which are VLC with regular construction and a specific code, also called Context-Adaptive Variable Length Coding which is used to encode residual block data, *i.e.* the values of the coefficients obtained after the zigzag reordering of the block. The encoding of a macroblock, whether for an Intra slice or an Inter (P) slice, as illustrated in Figure 3 (grey blocks being only present in P slices encoding process) relies on deriving an encoding, with VLC tables that can depend on previously coded elements, different prediction values as well as the block residual values (the most complex part in the H.264 encoding process).

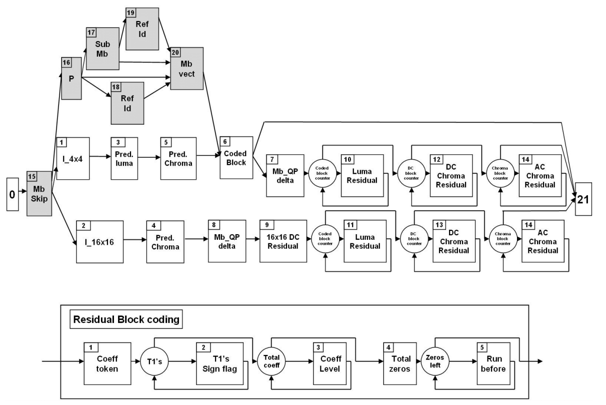

Figure 3 – Macroblock decoding process for I/P slices in H.264 standard.

B. Bits 'selected for encryption' in H.264/AVC codestreams

To better illustrate and explain the concept introduced in Section II.C, let us consider a detailed example, namely the ciphering of the codeword providing information on the Mb_QP_Delta parameter, that is to say the symbol changing the value of the quantification parameter (QP) in the macroblock layer. The value of *Mb_QP_delta*, coded by a signed Exp-Golomb codeword, shall be in the range [-26; +25]. An example of a signed Exp-Golomb code with 9 codewords is given in Table 1.

Table 1 - Example of a signed Exp-Golomb code with 9 codewords

Index	Codeword	Mb_QP_ Δ
0	1	0
1	010	1
2	011	-1
3	00100	2
4	00101	-2
5	00110	3
6	00111	-3
7	0001000	4
8	0001001	-4
…	…	…

Noticing that the change of the Mb_QP_Delta has no influence on the rest of the decoding, only the risk of desynchronization rules out some configurations: it is then proposed to mark as 'selected for encryption' the suffix bits (*i.e.* the bits following the first '1'), as illustrated by the highlighting of the corresponding bits in Table 1.

Following this example, other bits can easily be 'selected for encryption' in the codewords for the Intra 4x4 prediction mode, the Macroblock type, the Intra chroma prediction mode, trailing ones, coefficient level suffix, total zeros, run before, Reference id or motion vectors… while always keeping in mind that before marking any bit as 'selected for encryption', one shall check if the condition on standard compliance will be verified when changing those bits. In particular, blocks located on the borders of the video slice won't be candidate for ciphering as they do not admit all prediction modes.

IV. NUMERICAL RESULTS

Results obtained with the proposed partial ciphering method are shown in Figure 4-Figure 6 for different quantification levels, different sequences and different formats. Those results have been obtained by ciphering various bits following the application of the partial encryption mechanism to H.264/AVC, with verification model JM 8.4. In practice, this corresponds to ciphering roughly about 25% of the Intra slices, and about 10-15% of the P slices, the percentage difference being mainly due to the fact specific Inter slices symbols such as Mb vect or Ref-Id (see Figure 3) were not considered in the 'selection for encryption' process.

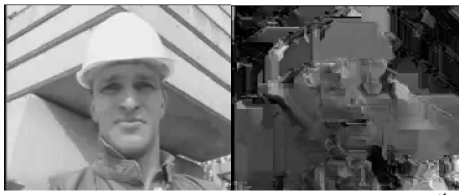

Figure 4 - Visual results for de-ciphered and ciphered 1st frame of 'Foreman' sequence (I coded), QCIF format, QP=30.

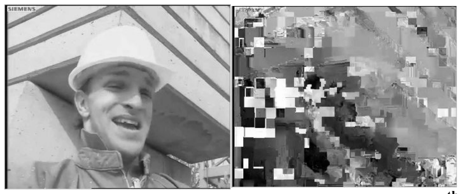

Figure 5 - Visual results for de-ciphered and ciphered 115th frame of 'Foreman' sequence (P coded), CIF format, QP=30.

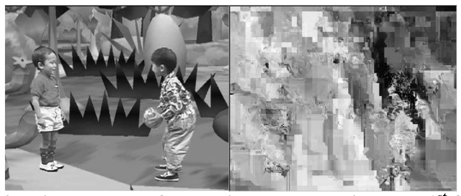

Figure 6 - Visual results for de-ciphered and ciphered 1st frame of 'Children' sequence (I coded), CIF format, QP=15.

The comparison between the de-ciphered sequences (on the left) and the ciphered ones, corresponding to the video decoding without decryption shows empirically that in all cases, whether for CIF or QCIF format, for high quality images (low QP) or lower quality images (high QP), for different sequences ('Foreman' or 'Children') and both for intra (I) and predicted images (P), the proposed ciphering method provides an encrypting level that is visually satisfying. This is confirmed from a more objective point of view by the PSNR evolution presented in Figure 7 for the 'Foreman' sequence in QCIF format, QP=30, intra refresh rate of one I every 15 images (IP_{14}) with a sequence length of 255 frames. A degradation of about 25 to 30 dB is observed for the Y component, and of about 10 dB for the U and V components, this difference being due to the fact that chrominance components are less encrypted is this scheme, as they have less importance in the visual restitution.

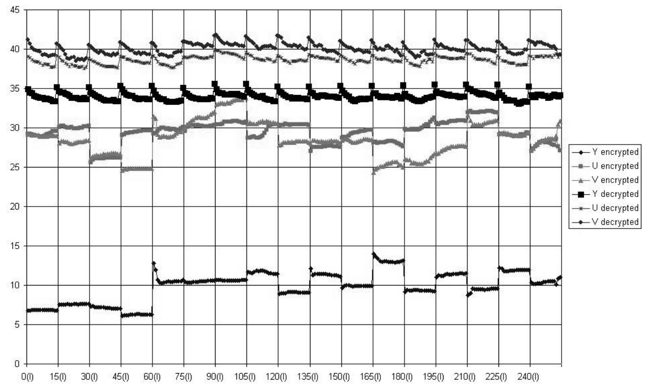

Figure 7 – Y, U and V components PSNR evolution of decrypted and non-decrypted decoded 'Foreman' sequence (Qcif, QP=30, 255 frames, IP_{14}).

V. CONCLUSIONS

The embedding of a partial ciphering mechanism for video standard is considered in this paper, where an original algorithm for selecting bits to be ciphered is proposed. This proposed algorithm is fully compatible with the H.264/AVC standard, and simulation results show that its application leads to PSNR degradations of about 25 to 30 dB when the ciphering key is unknown, making the video unrecognizable.

This method can be used in any situation where the content provider or user wants to keep confidentiality of the transmitted content against a potential intruder, with the added advantage that even the most simple (and basics) decoders will not crash when trying to decode the encrypted bitstream but also that the respect of the standardized video format can not help intruders to cryptanalyze the transmitted data.

ACKNOWLEDGEMENT

This work was partially supported by the European Community through project IST-FP6-001812 PHOENIX.

REFERENCES

[1] B.M. Macq and J-J. Quisquater "Cryptology for digital TV broadcasting," Proceedings of the IEEE, 83(6), pp. 944-957, June 1995.

[2] H. Cheng and X. Li, "Partial encryption of compressed images and videos," IEEE Trans. on Signal Processing, vol. 48, n. 8, August 2000.

[3] ITU-T Recommendation H.264, Geneva, May 2003.

[4] FIPS 197 (AES) http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf NIST, FIPS PUB 197, Advanced Encryption Standard, Nov. 2001.

[5] SP 800-38A http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf Recommendation for Block Cipher Modes of Operation - Methods and Techniques, Dec. 2001

[6] L. Tang, "Methods for encrypting and decrypting MPEG video data efficiently", Proceedings of the ACM Multimedia'96, pp. 219-229, Boston, USA, Nov. 1996.

[7] A.M. Alattar, G.I. Al-Regib and S.A. Al-Semari, "Improved selective encryption techniques for secure transmission of MPEG video bit-streams", Proceedings of ICIP'99, vol. 4, pp.256-260, 24-28 oct. 1999.