FINE GRANULARITY SCALABILITY ENCRYPTION OF MPEG-4 FGS BITSTREAMS

Bin B. Zhu[1], Yang Yang[2], Chang Wen Chen[3], Shipeng Li[1]
[1]Microsoft Research Asia, Beijing 100080, China
[2]Dept. of Elec. Eng. & Info Sci., Univ. of Sci. & Technol. of China, Hefei, Anhui 230027, China
[3]Dept. of Elec. and Computer Eng., Florida Inst. of Technology, Melbourne, Florida 32901, USA
[1]*{binzhu,spli}@microsoft.com,* [3]*cchen@fit.edu*

ABSTRACT

In this paper, we present an encryption scheme for MPEG-4 FGS which provides the same or a little coarser granularity of scalability after encryption. The scheme encrypts compressed data of each video packet or block independently. Initialization vectors are generated with a method to minimize the overhead. The scalability provided in an encrypted codestream using this scheme enables intermediate nodes to truncate an encrypted bitstream at near R-D optimality directly without decryption, which enhances system security. The scheme has virtually negligible overhead, and produces encrypted codestream with virtually the same error resilience performance as the unencrypted case. These features are very desirable in many applications.

1. INTRODUCTION

MPEG-4 has recently adopted a streaming video profile [1][2] which provides Fine Granularity Scalability (FGS) so that a single codestream can be easily truncated near rate-distortion (RD)-optimally to fit a wide variety of applications. MPEG-4 FGS compresses a video sequence into a base layer and an enhancement layer. The base layer is a non-scalable coding of the sequence at the lower bound of a bitrate range. The enhancement layer encodes the residue between the original sequence and the reconstructed based layer in a scalable manner to offer a wide range of bitrates within a single codestream. Temporal scalability is also supported. The coding efficiency of MPEG-4 FGS has been improved substantially with recent progress such as the scheme proposed in [3].

For many applications, video content encoded with MPEG-4 FGS needs to be protected against unauthorized consumption. MPEG-4 has recently adopted a Digital Rights Management (DRM) framework, eXtensions to the Intellectual Property Management and Protection (IPMP-X) [4] for MPEG-4 codestreams. Encryption is typically used in such a DRM system. A very challenging requirement in the design of scalable codestream encryption is that the functionalities such as FGS should not be impaired after encryption. For example, an MPEG-4 FGS codestream can be easily truncated to fit varying transmission bandwidths or display characteristics of a variety of devices. An encrypted MPEG-4 FGS codestream should preserve the original fine or at least maintain coarser granularity scalability such that an intermediate node in the content delivery path can still perform necessary truncation near R-D optimally on the encrypted codestream without decryption. This would enhance the end-to-end system security since many intermediate nodes may need to truncate an encryption codestream throughout its life. Scalable encryption with FGS enables these nodes to perform the required processing without sharing the secrets. It also improves error resilience when an encrypted scalable codestream is transmitted over a lossy or error-prone network since scalable encryption offers better synchronization and segmentation which reduce error expansion.

Many encryption schemes for FGS scalable codestreams such as JPEG 2000 and MPEG-4 FGS have been proposed [5]-[15]. Most of these schemes are reviewed in [16]. One scheme called SMLFE [9][10] encrypts video data in each Video Packet (VP) with the C&S encryption [18] for MPEG-4 FGS. The scalable granularity is reduced to a VP-level after encryption. Any ciphertext bit error in a VP renders the whole decrypted VP unusable, thanks to the dependency on the whole ciphertext for the C&S encryption. This scheme may not fit some applications which require truncation at a level smaller than a VP or better error resilience is required. In this paper, we propose an encryption scheme for MPEG-4 FGS to address these issues yet maintaining virtually negligible overhead. This scheme has two operational modes: VP Encryption (VPE) and Block Encryption (BE) modes. In the VPE mode, compressed data in each VP is independently encrypted without emulating the VP delimiters or increasing size. In the BE mode, compressed enhancement data of each 8 by 8 block or macroblock (MB) is independently encrypted bitplane-wise from the Most Significant Bit (MSB) to the Least Significant Bit (LSB). The ciphertext is partitioned and allocated to each video packet, with stuffing bits possibly inserted to avoid delimiter emulation. The former method enables truncation at a VP or trailing VP level, and the latter method preserves the full scalability of a scalable bitstream. Both have an error resilience performance virtually the same as the unencrypted codestream. The base layer which does not provide any scalability is always encrypted with VPE mode. The scheme is applicable to other scalable coding such as the one in [3].

This paper is organized as follows: MPEG-4 FGS and a syntax compliant encryption are briefly described in Section 2. The detail of our proposed scheme is described in Section 3, with discussions on its performance in Section 4. Experimental results are reported in Section 5 and we conclude the paper in Section 6.

2. PRELIMINARIES

2.1. MPEG-4 FGS

In MPEG-4, a *Video Object Plane* (VOP) is an instance at a given time of Video Object (VO), an entity in the codestream that can be accessed and manipulated. MPEG-4 FGS encodes a video sequence into two layers: a base layer which is encoded with a non-scalable coder to provide the lowest quality and bitrate for a scalable codestream, and a scalable enhancement layer which offers enhancement to the base layer. More precisely, the difference between the original VOP and the reconstructed VOP from the base layer is encoded bitplane-wise

from MSB to LSB. Each bitplane of a block's DCT coefficients is zigzag-ordered, converted to (RUN, EOP) symbols, and coded with variable-length coding to produce enhancement layer codestream, where RUN is the number of consecutive zeros before a nonzero value and EOP indicates if there is any non-zero value left on the current bitplane for the block. For FGS Temporal scalability (FGST) which does not have corresponding base layer VOPs, the bitplane coding is applied to the entire DCT coefficients of the VOP. MPEG-4 FGS provides very fine grain scalability to allow near RD-optimal bitrate truncation in a large range of bitrates. Auxiliary side information such as RD-data can be used to help such truncations to fit different application scenarios [3].

MPEG-4 FGS groups video data into Video Packets (VPs). Each VP is delimited by unique resynchronization markers to stop error propagation to other VPs. Necessary information is inserted after a resynchronization marker to enable resuming decoding. For the enhancement layer, both the bitplane start marker, *fgs_bp_start_code*, and the resynchronization marker *fgs_resync_marker* are used as VP delimiters. *fgs_bp_start_code* has 32 bits, starting with 23 binary zeros followed by 0xA and five bits to indicate which bit-plane the data belongs to. The marker *fgs_resync_marker* is 22 binary zeros followed by a binary one. The number of the first MB is inserted after each *fgs_resync_marker*. VP boundary is aligned with that of an MB. If a bit error occurs in a compressed bitplane data, the bitplane data of the current and subsequent blocks cannot be correctly decoded, and will be dumped. The lower bitplane data of those affected blocks are also dumped since the sign bits inserted in the undecodable bitplane data are lost which causes misalignment and wrong decoding of lower bitplanes. The nominal size of VP is determined at encoding time for targeted application scenarios. Smaller VP is used when the targeted application environment is error-prone such as transmission over wireless channels.

In practical applications, both layers are typically protected unequally against transmission imperfection. The base layer is usually well protected against bit errors or packet loss. The enhancement layer, on the other hand, is lightly or not protected.

2.2. Ciphertext Switch Encryption (CSE)

CSE is a syntax compliant encryption that would generate syntax-compliant ciphertext for any syntax-compliant plaintext. The ciphertext has the same size as the plaintext. CSE is divided into two stages. A conventional stream cipher is applied to plaintext in the first stage. A post-processing on the resulting ciphertext is followed to remove any syntax-offensive substreams. Details of CSE can be found in [17].

3. FGS SCALABLE ENCRYPTION SCHEME

To enable FGS truncation of a protected bitstream, encryption should be applied independently to small groups of data. Data grouping for MPEG-4 FGS encryption should reflect underlying encoded bitstream's error propagation characteristics to avoid further error expansion incurred by encryption. From the characteristics of MPEG-4 FGS described in last section, we choose VP or MB as the basic element for independent encryption. Our encryption scheme has therefore two operational modes: Video Packet Encryption (VPE) and Block Encryption (BE) modes.

3.1. Video Packet Encryption (VPE) Mode

Encryption in this mode is similar to SMLFE [9][10] that compressed video data in each VP is independently encrypted. VP headers and MB numbers are not encrypted. A syntax-compliant encryption scheme such as the Ciphertext Switching Encryption (CSE) [17] is used so that ciphertext would never emulate the VP delimiters. For MPEG-4 FGS enhancement layer, ciphertext does not contain any byte-aligned 22 consecutive binary zeros. VPE is also used in our scheme to encrypt base layer which does not provide any scalability.

This scheme offers several advantages over SMLFE. The overhead to identify VP delimiter emulation is avoided and truncation of trailing data in an encrypted VP is allowed in this scheme. SMLFE has to treat each VP as an atomic unit and no truncation is allowed to break a VP. The proposed scheme also has better error resilience than SMLFE. When an error occurs in ciphertext, all the blocks in the VP that the error occurs is garbled in SMLFE while only the current and subsequent blocks in the VP are garbled in the current scheme.

3.2. Block Encryption (BE) Mode

The scheme in this mode encrypts the compressed enhancement bitstream of each block independently from MSB to LSB with a conventional block cipher or stream cipher. The resulting ciphertext is then partitioned into smaller groups and allocated to each bitplane which is then packed into VPs. If a stream cipher such as RC4 [19] or SEAL [19] is used, then the grouping boundary is at the end of each bitplane of the block. In other words, the Contribution of a Block to a Bitplane (CBB) in this case contains the same number of bits as the unencrypted case.

Once a VP has all the CBBs from contributing blocks, the scheme checks encrypted data for VP delimiter emulation. If it finds a byte-aligned substream of 21 binary zeros, a binary one is inserted at the end of the substream. The proposed mechanism avoids emulation of the enhancement layer VP delimiters *fgs_bp_start_code* and *fgs_resync_marker* in the encrypted video data within a VP. At decryption side, the inserted bits are removed before decryption is applied. To do so, the encrypted data in a VP is scanned, if a byte-aligned substream of 21 binary zeros is found, then the following bit is dropped. This delimiter avoidance mechanism is more efficient than the delimiter emulation identification mechanism used in SMLFE [9][10].

Decryption in BE mode is executed along with checking Variable Length Codes (VLCs) for the end of a block. Once the end of a block is found, the decryption of the next block is used to decrypt the remaining data in a VP until end of the block is found. This process continues until all the encrypted data in a VP is decrypted to corresponding individual blocks. This process is possible since a stream cipher is used in the BE mode that there is a bit-to-bit correspondence between a ciphertext and its plaintext.

The block in the BE mode can be either a block of 8 by 8 or a macroblock (MB), depending on the granularity of scalability the encrypted codestream needs to support. In typical applications, MB is fine enough.

3.3. Initialization Vector (IV) Generation

An Initialization Vector (IV) is generated for each independent encryption in our scheme. Different IVs are used for encryption of different blocks or VPs. This avoids the cases

when the same random sequence generated by a stream cipher is applied to repetitively encrypt different blocks or VPs, or a stream cipher produces the same initial or whole output for two blocks or VPs when they have the same initial or whole data to be encrypted. This method is less efficient than the C&S encryption [18] used in SMLFE which achieves the same goal by applying a reversible bilinear "hash" function to the data to be encrypted to generate a data-dependent partial key which is combined with the content encryption key to encrypt the data. The gain in our case is a causal encryption[1] that encryption of subsequent data does not affect already encrypted data. This enables truncation of trailing ciphertext and there is only forward error expansion (i.e., an error affects only the current and subsequent decrypted data of a block or VP).

Each 8 by 8 block or MB can be uniquely identified by the block's index and color component for each VOP. In MPEG-4 FGS, the number of the first MB is inserted right after each *fgs_resync_marker*. The MB after *fgs_bp_start_code* is always the first MB. The last five bits in *fgs_bp_start_code* is used to identify the bitplane. Therefore the bitplane ID in *fgs_bp_start_code* and the number of the first MB in a VP can be used to uniquely identify a VP in a VOP. The unique identifier can be used to generate the IV used to encrypt a VP or block so that the IV for each independent encryption in our scheme is not inserted into the codestream which would otherwise cause a large overhead as in the scheme proposed in [6].

The actual IV generation mechanism depends on the format the codestream uses. If the codestream contains persistent presentation time or other attributes unique for each VOP, which is invariant throughout the life of the codestream, even when some VOPs are lost, then this unique VOP identification attribute is used to generate IVs for the VOP. More specifically, a random IV is inserted into a codestream as a global IV. This global IV is hashed together with the unique VOP identification attribute, layer ID (i.e., base layer or enhancement layer), and the unique VP or block identifier to generate the IV used to encrypt the VP or the block in base and enhancement layers.

If such a persistent VOP identifier does not exist, then an independent random IV (called VOP IV) is inserted for each VOP. This VOP IV is inserted to the base VOP for FGS VOP and FGST VOP itself otherwise. The VP or block identifier is combined with the layer ID which is used to identify base or enhancement layer, copy-expanded to the length of IV, and then is XORed with the VOP IV the VP or block belongs to. The result is used as IV or seed for a stream cipher to encrypt the VP or block. In case of block ciphers used for encryption, the XORing result is encrypted with the block cipher, and the output is used as the IV to encrypt the VP or block.

4. PERFORMANCE AND DISCUSSION

Our scheme has virtually negligible overhead. If there exists a persistent VOP identifier in the format that a codestream is packed into, then the overhead is a single IV for the whole codestream in the VPE mode. In the BE mode, there exists additional overhead from stuffing bits that may be inserted into an encrypted bitstream to avoid emulation of the VP delimiters. Stuffing bit insertion occurs very rarely. The probability is roughly about $1/2^{21}$. If such a persistent VOP identifier does not exist, a VOP IV is inserted to each VOP. This overhead is 960 bps if the frame rate is 30 frames per second. This translates into 0.048% if an MPEG-4 FGS codestream is compressed at 2.0 Mbps. Even if the codestream is truncated to 100 Kbps, this overhead is only 0.96%.

When an error occurs in ciphertext, decryption of subsequent data of the block or VP that the error occurs is affected (if the rare case of minor backward expansion is ignored when CSE is used). This behavior is the same as the unencrypted case. Therefore our scheme has virtually the same error resilience performance as the corresponding unencrypted codestream.

The VPE mode provides VP-level or trailing VP data truncation in an encrypted codestream. The BE mode enables block-level truncation. Auxiliary side information such as RD-data can be used for such level truncations. This is the same as the unencrypted case as described in [3]. Therefore the BE mode provides the same scalability as the unencrypted codestream. An encrypted codestream in the BE mode can be truncated near optimally directly without decryption.

The two modes in our scheme are designed for different applications. In addition to the difference in granularity of scalability, VPE mode allows separation of encryption or decryption process from encoding or decoding process, and can be used with in situ encryption. PE mode is more complex and is coupled with encoding/decoding process.

5. EXPERIMENTAL RESULTS

We have implemented the proposed scheme based on the MPEG-4 reference software [21] and Crypto++ library [22]. For all the standard QCIF video sequences we tested, the base layer was coded at 30 frames per second with a nominal bitrate 90 kbps, while the enhancement layer was around 2.1 Mbps. All the experimental results were conducted on a DELL Precision 330 PC with P4 CPU of 1.40 GHz and 512 MB of RAM. CSE with RC4 was used in the VPE mode. RC4 was used in the BE mode. An IV is used as a partial key for each independent encryption.

The encryption speed for the VPE mode was tested. In this mode, encryption is applied directly to the output buffer without much of other processing overhead. The experimental results are shown in Figure 1. Note that CSE has the same speed for encryption and decryption. When the nominal VP size was set at 400 bits, the processing speed of CSE with RC4 was around 7.70 MBps. The encryption speed increases with the VP size. CSE adds negligible processing overhead to the underlying stream cipher RC4. A separate experiment showed that setting up a key in RC4 was very slow. It took substantial percentage of the overall processing time at a small VP size. A larger VP size reduces the number of VPs, i.e., the number of setting-up RC4's keys, and improves the CSE's processing speed. Other stream ciphers with more efficient key setup can also improve CSE's processing performance.

[1] CSE may affect already encrypted data when an illegal symbol occurs in ciphertext. Such an occurrence is very unlikely. If it does occur, backward error expansion is usually confined within the last encrypted symbol.

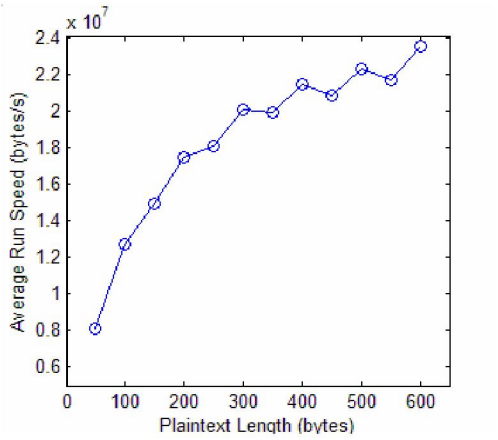

Figure 1: Encryption speed in the VPE mode for various VP data lengths where CSE is used with SEAL.

We have also tested the overhead for the case that a 32-bit IV used for encryption was inserted for each frame. The overhead is virtually due to this IV insertion at 960 bps. The overhead due to insertion of stuffing bits to avoid emulation of the VP delimiters in the BE mode was difficult to be observed.

Error resilience was also tested informally for CSE with RC4 in the VPE mode and RC4 in the BE mode. In this experiment, error bits were randomly generated. The bits in both encrypted and unencrypted codestreams corresponding to those error bits were flipped. The two reconstructed sequences were shown side by side. We did not observe any difference in visual quality.

6. CONCLUSION

We presented an encryption scheme with two operational modes for MPEG-4 FGS and similar scalable coding schemes. In the Video Packet Encryption (VPE) mode, compressed data in each VP is independently encrypted by ciphertext switching encryption which produces ciphertext without emulation of the VP delimiters. In the Block Encryption (BE) mode, compressed enhancement data of each 8 by 8 block or macroblock is independently encrypted bitplane-wise from the most significant bit to the least significant bit. The result is partitioned and allocated to each video packet, with stuffing bits possibly inserted to avoid delimiter emulation. VPE enables truncation at VP or trailing VP, while PE preserves the full scalability of the encrypted codestream. Both modes have virtually the same error resilience performance as the unencrypted case, and introduce virtually negligible overhead. The presented scheme can be applied in a variety of applications.

REFERENCES

[1] *Coding of Audio-Visual Objects, Part-2 Visual, Amendment 4: Streaming Video Profile*, ISO/IEC 14496-2/FPDAM4, July 2000.

[2] W. Li, "Overview of Fine Granularity Scalability in MPEG-4 Video Standard," *IEEE Trans. on Circuits and Systems for Video Technol.*, vol. 11, no. 3, pp. 301–317, March 2001.

[3] F. Wu, H. Sun, G. Shen, S. Li, Y.-Q. Zhang, B. Lin, and M.-C. Lee, "SMART: An Efficient, Scalable, and Robust Streaming Video System," *EURASIP J. Appl. Signal Proc.* 2004:2, pp. 192–206, 2004.

[4] *Information Technology – Coding of Audio-Visual Object – Part 13: Intellectual Property Management and Protection (IPMP) Extensions*, ISO/IEC JTC1/SC29/WG11 14496-13:2004(E), 2004.

[5] R. Grosbois, P. Gerbelot, and T. Ebrahimi, "Authentication and Access Control in the JPEG 2000 Compressed Domain," *Proc. SPIE 46th Annual Meeting, Appl. of Digital Image Proc. XXIV*, San Diego, California, 2001.

[6] S. J. Wee and J. G. Apostolopoulos, "Secure Scalable Streaming Enabling Transcoding Without Decryption," *IEEE Int. Conf. Image Proc.*, Thessaloniki, Greece, vol. 1, pp. 437–440, Oct. 2001.

[7] M. Wu and Y. Mao, "Communication-friendly Encryption of Multimedia," *IEEE Workshop on Multimedia Signal Processing*, pp. 292–295, Dec., 2002.

[8] H. H. Yu, "Scalable Encryption for Multimedia Content Access Control," *IEEE Int. Conf. on Acoustics, Speech, and Signal Proc.*, vol. 2, pp. II-417–420, April 6-10, 2003.

[9] C. Yuan, B. B. Zhu, M. Su, X. Wang, S. Li, and Y. Zhong, "Layered Access Control for MPEG-4 FGS Video," *IEEE Int. Conf. Image Processing*, Barcelona, Spain, Sept. 2003, vol. 1, pp. 517 – 520.

[10] B. B. Zhu, C. Yuan, Y. Wang, S. Li, "Scalable Protection for MPEG-4 Fine Granularity Scalability," *IEEE Trans. on Multimedia*, vol. 7, no. 2, pp. 222-233, April 2005.

[11] H. Wu and D. Ma, "Efficient and Secure Encryption Schemes for JPEG2000," *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, 2004 (ICASSP '04)*, vol. 5, pp. V869 — 872, May 2004.

[12] Y. Wu and R. H. Deng, "Compliant Encryption of JPEG2000 Codestreams," *IEEE. Int. Conf. on Image Processing 2004 (ICIP'04)*, Singapore, Oct. 2004.

[13] B. B. Zhu, Y. Yang, and S. Li, "JPEG 2000 Encryption Enabling Fine Granularity Scalability without Decryption," to appear in *ISCAS 2005*.

[14] B. B. Zhu, M. Feng, and S. Li, "A Framework of Scalable Layered Access Control for Multimedia," to appear in *IEEE Int. Symp. Circuits and Systems 2005*.

[15] B. B. Zhu, Y. Yang, and S. Li, "JPEG 2000 Syntax-Compliant Encryption Preserving Full Scalability," to appear in *IEEE Int. Conf. Image Processing 2005*.

[16] B. B. Zhu, M. D. Swanson, and S. Li, "Encryption and Authentication for Scalable Multimedia: Current State of the Art and Challenges," *SPIE Conf. Internet Multimedia Management Systems V*, Philadelphia PA, Oct. 2004.

[17] B. B. Zhu, Y. Yang, and S. Li, "Ciphertext Switching Encryption," submitted for publication.

[18] M. H. Jakubowski and R. Venkatesan, "The Chain & Sum Primitive and Its Applications to MACs and Stream Ciphers," *EUROCRYPT'98*, pp. 281 – 293, 1998.

[19] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, 2nd ed., John Wiley & Sons, Inc. 1996.

[20] Federal Information Processing Standards (FIPS) Publication 197, *Advanced Encryption Standard (AES)*, Nov. 2001.

[21] *Information Technology – Codingof Audio-Visual Object – Part: Refereenmce Sofware*, ISO/IEC JTC1/SC29/WG11/N4711, March, 2002.

[22] *Crypto++*, http://www.eskimo.com/~weidai/cryptlib.html.