JPEG 2000 SYNTAX-COMPLIANT ENCRYPTION PRESERVING FULL SCALABILITY

Bin B. Zhu[1], Yang Yang[2], 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
[1]binzhu@ieee.org, [2]wdscxsj@ustc.edu, [1]spli@microsoft.com

ABSTRACT

An efficient syntax-compliant encryption scheme for JPEG 2000 and motion JPEG 2000 is proposed in this paper. Compressed visual data is completely encrypted yet the full scalability of the unencrypted codestream is completely preserved to allow near RD-optimal truncations and other manipulations securely without decryption. Compared with other reported schemes, our scheme shows advantages on syntax compliance, compression overhead, scalable granularity, and error resilience. In addition to preserving the original scalability, a JPEG 2000 codestream encrypted with our scheme has the same error resilience capability as the unencrypted codestream. The encrypted codestream is still syntax-compliant so that an encryption-unaware decoder can still decode the encrypted codestream, although the decoded visual data is completely garbled and meaningless. Our scheme has virtually no adverse impact on the compression efficiency.

1. INTRODUCTION

JPEG 2000 (J2K) [1] is a latest still image coding standard which provides high compression efficiency, lossy to lossless coding, and flexible scalability. A J2K codestream is organized in a hierarchical structure with packets as the fundamental building blocks. A J2K codestream provides Fine Granularity Scalability (FGS): it can be truncated to the preset layers (i.e. qualities), resolutions, components, tiles, etc., or to coding passes inside a packet to fit a large variety of application scenarios. FGS of a J2K codestream allows near Rate-Distortion (RD)-optimal bitrate reduction for a large range of rates. Motion JPEG 2000 which encodes each video frame independently is also defined [2].

Image protection is an important issue in many applications, and therefore is addressed by the JPEG 2000 specifications part 8, commonly known as JPSEC. JPSEC provides a framework that different protection technologies can be applied to. A recent paper [3] provides a brief review of JPSEC. This paper focuses on encryption for JPSEC.

A particular requirement for encryption of scalable codestreams is that an encrypted codestream should preserve as fine as possible granularity scalability so that it can be truncated directly by a potentially untrustworthy party without decryption. Otherwise the system security may be sacrificed. Encryption technologies to meet this requirement have been reviewed recently in [4]. Among them, some are specifically designed for JPEG 2000 encryption. Grosbois et al. [5] proposed two encryption schemes to allow accesses to resolutions and layers, respectively, without decryption, but the two types of accesses cannot be supported with a single encrypted codestream. Another drawback is that a seed used in encrypting a code-block is inserted after the termination marker of the code-block, and may be lost during transmissions or scheme-unaware truncations, resulting in an undecryptable code-block. Wee et al. [6] proposed a Secure Scalable Streaming (SSS) scheme which groups J2K packets into SSS packets. Data except header fields in each SSS packet is independently encrypted with a block cipher in Cipher Block Chaining (CBC) mode. The Initialization Vector (IV) used in encryption of each SSS packet is inserted into an unencrypted header of the SSS packet. Scalable granularity is reduced to a progressive SSS packet level. To reduce encryption overhead due to IVs, the number of SSS packets is not high, resulting in very coarse granularity scalability in an SSS encrypted codestream. We have proposed a code-block based encryption scheme [7] which preserves original FGS with small, about 1%, overhead on the compression efficiency.

A desirable feature for JPEG 2000 encryption is that an encrypted J2K codestream is J2K syntax-compliant that encrypted data does not emulate any delimiters to avoid erroneous parsing or synchronization, esp. under error-prone transmissions. All the above schemes do not meet this requirement. Wu and Ma [8] proposed a packet-level syntax-compliant encryption scheme which encrypts each byte by adding a pseudo-random byte modulo 0xFF. Wu and Deng [9] proposed to encrypt each Codeblock Contribution to a Packet (CCP) with a modular addition or a block cipher repetitively until the ciphertext is syntax-compliant. It is unclear how these two schemes deal with seeds or IVs, and therefore there is no way to estimate their overheads on the compression efficiency.

In this paper, we propose a scheme based on our previous scheme [7]. The scheme encrypts each codeword segment or each intersection of a codeword segment with a CCP independently with a syntax-compliant encryption primitive after J2K compression. The IV for each independent encryption is generated from a global IV and the unique index to the data to be encrypted. There is no need to store each IV. This scheme shows four major improvements over our previous scheme reported in [7]: 1) The encrypted codestream is now J2K syntax-compliant. 2) The overhead on the compression efficiency is virtually removed. 3) Tile-based cropping can be applied directly without decryption. 4) The scalable granularity after encryption is finer -- as fine as the unencrypted codestream. Our scheme also shows advantages on syntax compliance, compression overhead, scalable granularity, and error resilience over other reported schemes.

The rest part of this paper is organized as follows. JPEG 2000 is briefly introduced in the next section to provide a basis to describe our proposed scheme, which is described in detail in Section 3, along with comparison with other proposed J2K encryption schemes. Experimental results are presented in Section 4. We conclude our paper in Section 5.

0-7803-9134-9/05/$20.00 ©2005 IEEE

2. JPEG 2000 AND JPSEC

In J2K, an image can be partitioned into smaller rectangular regions called *tiles*. Each tile is encoded independently. Data in a tile are divided into one or more components in a color space. A wavelet transform is applied to each tile-component to decompose the image data into different resolution levels. The lowest frequency subband is referred to as the resolution level 0 subband, which is also resolution 0. The image at resolution r (r>0) consists of the data of the image at resolution (r-1) and the subbands at resolution level r. Wavelet coefficients are quantized by a scalar quantization to reduce precision of the coefficients except in the case of lossless compression. Each subband is partitioned into smaller non-overlapping rectangular blocks called *code-blocks*. Each code-block is independently entropy-encoded from the most significant bit-plane to the least significant bit-plane to generate an embedded bitstream. Each bit-plane is encoded within three sub-bitplane passes. In each coding pass, the bit-plane data and the contextual information are sent to an adaptive arithmetic encoder for encoding. By default, arithmetic coding is terminated at the end of the last bit-plane encoding, and a code-block's embedded bitstream forms a single *Arithmetic Codeword Segment* (ACS). J2K also allows termination at the end of each sub-bitplane coding pass that the bitstream from each coding pass forms an ACS. Context probabilities can also be re-initialized at the end of each coding pass to enable independent decoding of the bitstream from each coding pass. The optional arithmetic coding bypass puts raw bits into bitstream for certain coding passes. In this case, the boundary between arithmetic coding passes and raw passes must be terminated. Both ACS and raw codeword segment are referred to as *Codeword Segment* (CS) in this paper.

A code-block's bitstream is distributed across one or more layers in the codestream. Each layer represents a quality increment. A layer consists of a number of consecutive bit-plane coding passes from each code-block in the tile, including all subbands of all components for that tile. J2K also provides an intermediate space-frequency structure known as a *precinct*. A precinct is a collection of spatially contiguous code-blocks from all subbands at a particular resolution level. The fundamental building block in a J2K codestream is called a *packet*, which is simply a continuous segment in the compressed codestream that consists of a number of bit-plane coding passes from each code-block in a precinct. Data length of each CCP is indicated in the packer header. In the case of multiple codeword segments, the length of each CS in a CCP is indicated in the packer header. Each ACS or CCP does not allow byte-aligned value between 0xFF90 and 0xFFFF for any two consecutive bytes or ending with a byte of value 0xFF. A raw codeword segment when arithmetic coding bypass is enabled does not allow any byte-aligned nine consecutive bits of 1 or ending with a byte of value 0xFF. J2K uses the unattainable range of two consecutive bytes to represent unique markers to facilitate organization and parsing of the bitstream and to improve error resilience. Each packet can be uniquely identified by the five parameters: tile, component, resolution level, layer, and precinct. Each code-block can be uniquely identified by the following parameters: tile, component, resolution level, precinct, subband, and the coordinates of the upper left point of the code-block on the reference grid. Packets for a tile can be ordered with different hierarchical ordering in a J2K codestream by varying the ordering of parameters in nested "for loops", where each "for loop" is for one of the parameters uniquely specifying a packet. Details on J2K can be found in [1][10], and motion JPEG 2000 in [2].

JPSEC introduce two new marker segments. One is SEC in the main header which is used to carry overall information about the security tools and parameters applied to the image. The other is INSEC placed in the bitstream to provide information of localized security tools and parameters. Details on JPSEC are found in [11].

3. SYNTAX-COMPLIANT ENCRYPTION FOR JPEG 2000 AND MOTION JPEG 2000

Instead of encrypting each code-block independently as used in our previous scheme [7], this scheme encrypts each codeword segment independently after J2K compression. This change has the advantage that the boundary of encryption coincides with that of arithmetic coding, and the original scalability and error resilience are fully preserved after encryption. The syntax-compliant encryption primitives [12][13] we proposed recently are used in the scheme to ensure syntax compliance. The IV for each independent encryption is generated from a global IV and the unique index to the data to be encrypted. There is no need to store each IV. Therefore a major overhead is removed. Each code-block is partitioned into CCPs in such a way that the remaining CCPs can still be decrypted correctly whatever CCP the bitstream terminates at. An alternative scheme is to encrypt each intersection of a CCP with a codeword segment independently, with the advantage that each CCP can be encrypted in situ, a desirable feature if encryption is applied after compression. The codeword encryption scheme will be referred to as the normal scheme while the intersection encryption scheme is referred to as the alternative scheme. Details are described in the rest of this section.

3.1. Encryption and Decryption Primitives

While all the syntax-compliant encryption methods [12][13] we recently proposed can be applied, we choose the Ciphertext Switching Encryption (CSE) for stream cipher based encryption and Locally Iterative Encryption (LIE) for block based encryption in this paper. In CSE, illegal substrings are switched back to the plaintext substrings. CSE is applied to our encryption without any change. In LIE, plaintext is divided into blocks and each block is encrypted iteratively until the block's output is compliant. Boundaries of blocks are taken care of by LIE so that each block can be decrypted correctly. When LIE is used, a block cipher in Cipher Block Chaining (CBC) mode is used to encrypt full blocks, and the same block cipher in Cipher Feedback (CFB) mode is used to encrypt the last partial block, if applicable, with the register initialized with the ciphertext of last full block or IV if there is no full block. In this way, LIE with a block cipher can be applied to encrypting plaintext of any length into ciphertext of exactly the same length. Interested readers are referred to [12] for the detail of CSE and [13] for LIE.

3.2. IV Generation

A distinct IV is used for each independent encryption. The index to the code-block and the first coding pass of the encryption segment is used to generate this IV. A code-block can be uniquely identified by tile, component, resolution level, precinct, subband, and the coordinates of the upper left point of

the code-block on the reference grid, as we mentioned in Section 2. The coordinates of the upper right point on the reference grid are used to identify each tile and precinct in generating an IV. Due to invariance of these coordinates under cropping such as from aspect ratio of 16:9 to 4:3, an encrypted J2K codestream can be cropped by dropping some tiles and the resulting codestream is still fully decryptable.

A global IV is inserted in the SEC at the main header for an encrypted J2K codestream. For motion J2K, an independent random frame IV is inserted in each frame. This global IV, the unique code-block identifier, and the index to the first coding pass of the encryption segment are concatenated and cryptographically hashed, with possible truncation if necessary, to generate a unique IV to encrypt the current encryption segment.

In typical applications, the code-block identifier and the coding pass index can be represented together by a single word of length equal to IV. In this case, each individual IV can be generated simply by XORing the global IV with the word, and no hash operation is needed.

3.3. Syntax-Compliant JPEG 2000 Encryption

With the IV and encryption primitives, each segment after compression can be encrypted with the syntax-compliant primitives that the ciphertext is still syntax-compliant. On the player side, an encrypted J2K codestream is first decrypted and then decoded. The encryption process is straightforward if the alternative scheme is used, or if the normal scheme is used when each coding pass is terminated. In both cases, there is no compression overhead (SEC in JPSEC is not considered as overhead).

For the default case that the whole code-block is a single codeword segment, and if the normal scheme is applied, the ciphertext of an encrypted codeword segment may need to be partitioned into CCPs. Each CCP must be terminated at a right position that decryption can be executed correctly when the bitstream is truncated at the CCP, and the CCP cannot end with a byte of value 0xFF in either ciphertext or plaintext. This means that the original CCP partition points obtained without encryption may have to be modified after encryption is applied. For example, when LIE is used in CBC mode, a CCP has to terminate at a block boundary of the block cipher used in LIE. When CSE is used, a switched portion cannot be split into two consecutive CCPs. For CSE, if the last byte of the ciphertext is not 0xFF when the original partition (i.e., the partition obtained when encryption is not applied) is used (note that in this case the last byte of plaintext cannot be 0xFF either), then there is no change to the CCP boundary when encryption is applied, thus no compression overhead. Otherwise the boundary is moved to a following byte which is not of value 0xFF in either plaintext or ciphertext, resulting in compression overhead.

In CSE, switched consecutive bytes are typically two bytes long. This means that when moving is needed, CCP boundary is typically moved to the next byte, resulting in one byte overhead for the CCP. Since the chance that an encrypted CCP ends with a byte of value 0xFF is about 1 in 256, the overhead for CSE is very small, almost negligible. Due to this advantage, CSE is recommended when the normal scheme is used with the default arithmetic coding mode. For other cases both CSE and LIE can be equally applied without any overhead.

3.4. Scalability and Error Resilience

In addition to the aforementioned advantage of compression overhead, the scheme also has advantages on scalability and error resilience over other schemes. In our scheme, the data in a J2K codestream is fully encrypted, yet the full scalability of the unencrypted codestream is preserved. This flexibility is very desirable when a single codestream is used for a wide range of applications, esp. when some applications may not be known at encryption time. Wu and Ma's scheme [8] encrypts packet data in each packet. The scalable granularity is raised to packet-level after encryption. Except for trailing truncation, it is impossible to truncate to an arbitrarily selected CCP after encryption. Wu and Deng' scheme [9] encrypts each CCP, and the scalable granularity is raised to CCP-level. This is enough for many applications. It is not doable if an application wants to truncate at selected coding passes after encryption when each coding pass is terminated. Our scheme allows such fine truncations after encryption. Compared to other syntax-noncompliant encryption schemes such as our previous scheme reported in [7], this scheme generates encrypted, syntax-compliant codestream. An encryption-unaware decoder such as a decoder of an old version is still able to decode the encrypted codestream, although the decoded visual data is completely garbled and meaningless. Such a decoder may not be able to decode an encrypted codestream generated by a syntax-noncompliant encryption scheme.

An image or frame may be cropped. A widely used cropping in video is to convert the aspect ratio from 16:9 to 4:3. J2K enables such cropping without touching the compressed data by dropping tiles and adjusting some coordinates and parameters. A J2K codestream encrypted with our scheme enables such cropping too, thanks to the IV generation mechanism which generates invariant IV for each encryption segment under the cropping. Other schemes may also be able to do so if IVs are inserted in the bitstream, resulting in a high compression overhead. Our scheme does not have the overhead.

Errors may occur during transmission over networks. J2K offers several error resilience tools. When an error occurs, the current (i.e., where the error occurs) and subsequent coding passes in the same codeword segment may not be decompressed correctly and therefore dropped. Our scheme preserves this property too. When an error occurs in the ciphertext, the current and subsequent data in the codeword may not be decrypted correctly and therefore dropped. In other words, a J2K codestream encrypted with our scheme has exactly the same error resilience capability as the unencrypted codestream under all different J2K coding modes. Wu and Ma's scheme [8] may propagate errors to CCPs of other code-blocks. Wu and Deng' scheme [9] may propagate errors to preceding coding passes in the same CCP due to their global-level iterative encryption.

4. EXPERIMENTAL RESULTS

The proposed scheme has been implemented base on the publicly available J2K implementation JasPer [14]. Cryptographic primitives are based on Crypto++ [15]. SEAL [16] and AES [17] are used as the stream and block cipher primitives in CSE and LIE, respectively. Due to the length limitation, we report here only a couple of experiments. More results will be reported later in a full paper.

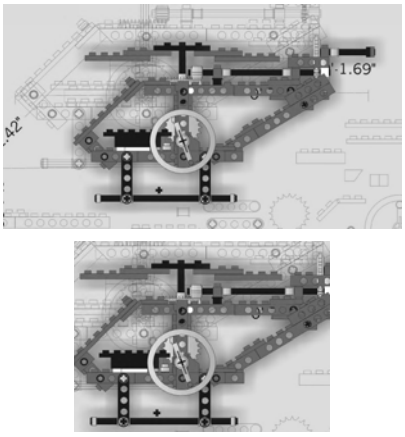

Figure 1: Cropping from aspect ratio 16:9 (1280 by 720 pixels) to 4:3 (792 by 594 pixels).

Figure 1 shows an encrypted J2K image cropped from aspect ratio of 16:9 to 4:3 directly without decryption. The coordinates used in this experiment are the same as the example shown in Section B.3 in [1].

Figure 2 shows encryption and decryption speeds for CSE (top), Wu and Ma's (left bottom two), and Wu and Deng's (left middle two) for typical CCP sizes. Note that CSE uses the same procedure for both encryption and decryption. There is only a single curve for CSE. It is clear that CSE is faster than the other two schemes. Lower speeds at short plaintext are due to the time spent on setting IV in SEAL. Setting IV in SEAL takes substantial portion of the overall time when plaintext is short.

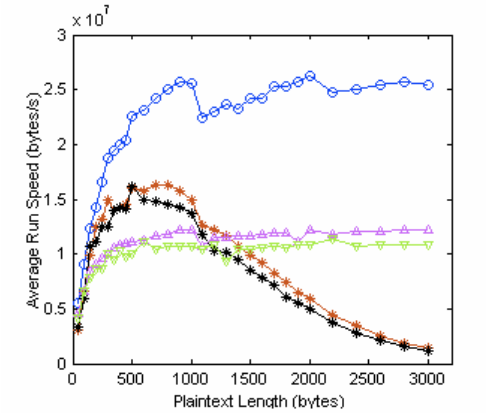

Figure 2: Encryption/decryption speeds for different syntax-compliant encryption schemes (see the main text for explanation).

5. CONCLUSION

We have proposed an improved encryption scheme from our early version for JPEG 2000 and motion JPEG 2000. The scheme produces JPEG 2000 syntax-compliant output. It offers several advantages over the old version and other schemes proposed in the literature. The scheme has virtually no overhead. Although the visual data is fully encrypted after compression, the original scalability and error resilience capability are fully preserved after encryption. Therefore an encrypted codestream can be truncated near RD-optimally or manipulated without decryption. The end-to-end system security has been boosted.

REFERENCES

[1] *Information Technology – JPEG 2000 Image Coding System, Part 1: Core Coding System*, ISO/IEC 15444-1:2000 (ISO/IEC JTC/SC 29/WG 1 N1646R, March 2000).

[2] *Information Technology – JPEG 2000 Image Coding System, Part 3: Motion JPEG 2000*, ISO/IEC 15444-3:2002.

[3] F. Dufaux, S. Wee, J. Apostolopoulos and T. Ebrahimi, "JPSEC for Secure Imaging in JPEG 2000," *SPIE Proc. Applications of Digital Image Processing XXVII*, vol. 5558, pp. 319-330, Nov. 2004.

[4] B. B. Zhu, M. D. Swanson, and S. Li, "Encryption and Authentication for Scalable Multimedia: Current State of the Art and Challenges," *Proc. SPIE Internet Multimedia Management Systems V*, vol. 5601, pp. 157-170, Philadelphia PA, Oct. 2004.

[5] R. Grosbois, P. Gerbelot, and T. Ebrahimi, "Authentication and Access Control in the JPEG 2000 Compressed Domain," *Proc. SPIE Appl. of Digital Image Processing XXIV*, vol. 4472, pp. 95-104, San Diego, California, Dec. 2001.

[6] S. J. Wee and J. G. Apostolopoulos, "Secure Scalable Streaming and Secure Transcoding with JPEG-2000," *IEEE Int. Image Processing*, vol. 1, pp. I-205-208, Sept. 1, 2003.

[7] B. B. Zhu, Y. Yang, and S. Li, "JPEG 2000 Encryption Enabling Fine Granularity Scalability without Decryption," *IEEE Int. Symp. Circuits and Systems*, Kobe, Japan, May 2005.

[8] 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.

[9] Y. Wu and R. H. Deng, "Compliant Encryption of JPEG2000 Codestreams," *IEEE. Int. Conf. on Image Processing 2004 (ICIP'04)*, pp. 3447-3450, Singapore, Oct. 2004.

[10] *JPEG2000 Verification Model 8.5 (Technical Description)*, ISO/IEC JTC 1/SC 29/WG 1 N1878, Sept. 2000.

[11] *JPSEC Commission Draft 2.0*, ISO/IEC/JTC 1/SC29/WG 1, N3397, 2004.

[12] B. B. Zhu, Y. Yang, and S. Li, "Ciphertext Switching Encryption," submitted for publication, 2005.

[13] B. B. Zhu, Y. Yang, and S. Li, "Locally Iterative Encryption," submitted for publication, 2005.

[14] JasPer, http://www.ece.uvic.ca/~mdadams/jasper.

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

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

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