JPEG 2000 Encryption Enabling Fine Granularity Scalability without Decryption

Bin B. Zhu, Shipeng Li
Microsoft Research Asia, Beijing 100080, China
{binzhu, spli}@microsoft.com

Yang Yang
Dept. of Elec. Eng. & Info Sci., Univ. of Sci. & Technol. of China, Hefei, Anhui 230027, China

***Abstract*—In this paper, we propose a novel encryption scheme for JPEG 2000 (J2K) and motion JPEG 2000. A block cipher in CBC mode is used to encrypt the bitstream of each J2K code-block. The encrypted J2K codestream preserves almost the same fine granularity scalability as the original J2K codestream yet with small or negligible overhead, and has fine and near RD-optimal truncations for a large range of bitrates. The proposed scheme enables desired transcoding directly on a single encrypted codestream without decryption to fit diverse capabilities of devices and heterogeneous networks with time-varying bandwidths. Any node, trusted or not, along the delivery path is able to perform desired transcoding without sacrificing the end-to-end security of the system.**

I. INTRODUCTION

JPEG 2000 (J2K) is a new image coding standard with fine granularity scalability (FGS) [1]. A J2K codestream is organized in a hierarchical structure with structural elements tiles, components, resolution levels, precincts, and layers. A packet is the fundamental building block in a J2K codestream, and is uniquely identified by the five aforementioned structural parameters. A J2K codestream provides FGS: the codestream can be truncated to the preset layers (i.e. qualities), resolutions, components, or to break packets to truncate at coding passes to fit a large variety of applications with devices of diverse capabilities and heterogeneous networks of different characteristics. FGS of a J2K codestream allows near Rate-Distortion (RD)-optimal bitrate reduction for a large range of bitrates. JPEG 2000 has also defined motion JPEG 2000 which encodes each video frame independently [2].

Content should be protected against unauthorized usage. This is achieved typically by encrypting the content and ensuring that only authorized users can access the decryption keys. Protection can be further refined that authorized users can only consume protected content according to the acquired rights. This is done with a Digital Rights Management (DRM) system which provides persistent protection for content from creation to consumption [3][4]. In either simple or DRM protection, a J2K codestream should be encrypted such that the encrypted codestream still preserves certain level of scalabilities, preferably the original FGS. Such scalability enables desired transcoding directly on an encrypted stream without decryption. Otherwise each intermediate processing node, possibly untrusted, along the delivery path, needs to access the encryption secrets to decrypt the encrypted content first, transcode to a desired stream, and then re-encrypt the resulting stream, which may dramatically lower the end-to-end security of the system.

Many multimedia encryption schemes have been proposed in the literature. Some are designed specifically for scalable streams. A comprehensive review on scalable multimedia encryption schemes, i.e., the schemes that preserve certain level of scalabilities in the encrypted stream, is given in [5]. As for JPEG 2000 encryption, Grosbois et al. [6] proposed two encryption schemes to provide access control on either resolutions or layers. To provide access control on resolutions, signs of wavelet coefficients in high frequency subbands are pseudo-randomly flipped. The output of a pseudo-random sequence generator is used to determine if the sign of a coefficient is inverted or not. A different seed to the generator is used for each code-block. Each seed is encrypted and inserted into the codestream right after the last termination marker of the corresponding code-block by exploiting the fact that any byte appearing behind a termination marker is skipped by a J2K standard compliant decoder. The resulting encrypted codestream is J2K format compliant. To provide access control on J2K layers, the bitstream of coding passes belonging to last layers are pseudo-randomly flipped in the same way as that used for image resolution scrambling. One drawback of the scheme is that the two types of access control cannot be supported with a single encrypted stream. Another drawback is that a seed inserted after the last termination marker of a code-block may be lost during truncation or transmission, rendering the code-block undecryptible. Wee et al. [7] proposed an encryption scheme called *Secure Scalable Streaming* (SSS) that works with J2K. The scheme groups J2K packets into SSS packets. All data except header fields in each SSS packet are independently encrypted with a block cipher in Cipher Block Chaining (CBC) mode. The Initialization Vector (IV) used in the encryption is inserted into the header of each SSS packet, which may add significant overhead, esp. if FGS is needed to be supported in the encrypted stream. Scalable granularity is also reduced to a progressive SSS packet level. The supported adaptations in SSS are either to drop an entire SSS packet or to truncate trailing data in a SSS packet. To reduce encryption overhead, the number of SSS packets for each J2K compressed image is not high, resulting in very coarse granularity of scalability in an SSS-encrypted stream. The paper [7] gives an example of 9 SSS packets, in either 3RX3L or 1RX9L setting that supports 3 resolutions and 3 layers, or 1 resolution and 9 layers, respectively. Other scalabilities are not supported since individual J2K packets cannot be directly accessed after SSS encryption. For example, an SSS stream in the 1RX9L setting does not support multiple resolutions.

0-7803-8834-8/05/$20.00 ©2005 IEEE.

In this paper, we propose a novel encryption scheme for JPEG 2000 that enables FGS in an encrypted J2K codestream yet with very small or negligible overhead. In our scheme, the bitstream of coding passes of each code-block, possibly padded with stuffing bits to ensure the length is a multiple of the encryption block size if ciphertext stealing [8] is not used, is independently encrypted with a block cipher such as AES in CBC mode. A single "global" IV is randomly generated and inserted into the image's header fields. The IV used for encrypting each code-block is generated by hashing the global IV along with the parameters that uniquely identifies the code-block. The ciphertext of each code-block is then partitioned into smaller blocks, each block is aligned with the encryption block size, and put into J2K packets. Unlike SSS, we don't use our own packets. The granularity of scalability after the encryption with our proposed scheme is nearly the same as the original JPEG 2000: an encrypted codestream can be truncated at any J2K packet, or each individual J2K packet can be reshaped by truncating trailing ciphertext of one or more code-blocks inside the packet. Auxiliary data for RD-optimal cutoff points can be inserted into header fields for near RD-optimal truncations in a large range of bitrates. For motion JPEG 2000, an independent random IV is generated for each frame and inserted into the frame's header. The same encryption scheme is applied to encrypt each frame.

This paper is organized in the following way: In the next section, we briefly introduce JPEG 2000 which is the basis to describe our proposed scheme. Our scheme is described in detail in Section III, along with comparison with other proposed J2K encryption schemes. Experimental results are presented in Section IV. We conclude our paper in Section V. Before we go to the next section, we would like to note that unless explicitly mentioned otherwise, a packet means a J2K packet and a header is not encrypted in this paper.

II. JPEG 2000

JPEG 2000 (J2K) is a wavelet-based image coding standard [1]. 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 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 a resolution r (r>0) consists of the data of the image at resolution (r-*1*) with 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. The coefficients in a code-block are 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. The arithmetic coding is terminated at the end of the last bit-plane encoding for a code-block. For error resilience, J2K also allows for termination of the arithmetic coded bitstream as well as the re-initialization of the context probabilities at each coding pass boundary to enable independent decoding of the bitstream from each coding pass. The compressed bitstream from each code-block 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*. A packet is simply a continuous segment in the compressed codestream that consists of a number of bit-plane coding passes for each code-block in the precinct. Data length of each code-block in a packet is indicated in the packer header. 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 six parameters: tile, component, resolution level, precinct, subband, code-block index. All packets of a tile can be ordered with different hierarchical ordering in a J2K codestream by varying the ordering of the parameters in nested "for loops", where each "for loop" is for one parameter from the above list. Details on J2K can be found in [1], and motion JPEG 2000 in [2].

III. FGS ENCRYPTION FOR JPEG 2000 & MOTION JPEG 2000

In our J2K FGS encryption scheme, a random IV is first generated and inserted into J2K header fields. This IV is referred to as a "global" IV for the image. The bitstream from each code-block is independently encrypted with a block cipher in CBC mode from the first coding pass of the most significant bit to the last coding pass of the least significant bit. A block cipher partitions a plaintext into blocks of the same length as the block size of the block cipher to be used, typically 64 or 128 bits. Such a block is referred to as *encryption block* in this paper. If the J2K bitstream of a code-block, referred to as plaintext, is not aligned with the encryption block size, the last partial block is padded with stuffing bits to a full block. These padding bits are overhead of our proposed encryption scheme, as well as the global IV which is of the same size as an encryption block.

The IV for encrypting a code-block is generated in the following way: a hash function such as SHA-1 [8] is applied to the global IV along with the parameters that uniquely identify the code-block. The resulting hash value is wrapped into blocks of the size of IV and XORed with each other. The result is used as the IV for encryption of the code-block. This code-block IV can be regenerated at decryption side and is *not* inserted into the codestream, which is very different from SSS proposed in [7].

After encryption, the ciphertext of each code-block is partitioned into smaller blocks which are all aligned with boundaries of encryption blocks and closest to the original partition if no encryption were used. These blocks of

ciphertext are then placed to packets of different layers in a similar way as the original J2K packetization. Auxiliary data for RD-optimal cutoff points can be inserted into header fields for near RD-optimal truncations in a large range of bitrates if packets are needed to be rate-reshaped.

An image is usually encoded and encrypted at the highest rate of a range of supported bitrates. An encrypted J2K stream can be truncated at a preset resolution, layer, and/or component determined at the packetization time during encryption. For example, to truncate an encrypted J2K stream to a certain layer, all packets of higher layers are dropped. To truncate to a certain resolution, all packets of higher solution levels are dropped. Please note that packet headers are not encrypted in our scheme. Therefore each packet can be easily identified and directly accessed in an encrypted codestream. In addition to packet level truncations, a packet can also be rate-reshaped if finer granularity of scalability is needed. In this case, the trailing ciphertext of each code-block inside the packet can be independently truncated. Such a truncation should be aligned with encryption block boundaries.

At receiver side, the IV for each code-block is regenerated from the inserted global IV. The ciphertext of each code-block in each received packet can be fully decrypted, thanks to the aforementioned packetization and truncation methods which generate encryption block aligned ciphertext for each code-block inside a packet. Since the J2K packet header includes information for the length of bitstream from each code-block in the packet, the ciphertext for each code-block in a packet can be easily identified. After decryption, the compressed bitstream for each code-block in a packet is decoded with an arithmetic decoder. Since ciphertext of a code-block is aligned with the encryption block size, the bitstream of a coding pass may be partitioned into two packets. Therefore the decrypted bitstream of a code-block in a packet may end with a partial coding pass. In this case, the data corresponding to a partial coding pass are also input to the arithmetic coder for decoding. Decoding of the code-block pauses and waits for more data when the current decrypted bitstream is exhausted. When the next block of data of the code-block arrives, decoding is resumed. In this way, the data from each received packet is all used for decoding. The overhead is just those last bits that cannot generate a complete decodable symbol. This overhead is negligible. Therefore, even though our scheme has a very small overhead after encryption, the resulting codestream after truncations has negligible overhead. This is very different from SSS which has larger overhead in terms of percentage when a smaller number of SSS packets are used at decryption and decoding.

Due to limitation of paper length, performance of the proposed scheme over a lossy communication network will not be discussed in this paper. It will be discussed in detail in a separate and lengthy paper. We only mention the result here: our encryption scheme has the same error resilience performance as the original J2K codestream if the error resilience option is not used. When the error resilience option is turned on, which results in a significant overhead, our scheme has a little worse error resilience performance than the original unencrypted J2K codestream.

Our scheme has a few advantages over SSS. Our scheme has much finer granularity of scalability in the encrypted stream than SSS yet with smaller overhead, esp. when truncation occurs, as mentioned above. Each J2K packet and data of coding passes from individual code-blocks in a packet can be directly accessed. Therefore our scheme can generate a single encrypted stream for diverse applications without having to use different encrypted streams as used in SSS. As shown in the next section, our scheme always generates fine, near RD-optimal truncations even though quite a few resolutions are simultaneously supported with the same encrypted codestream. In SSS, to support resolutions, less quality layers are supported to maintain the overhead at a fixed level, which results in very coarse, stairstep-like RD curves. As compared to the scheme proposed in [6], our scheme uses a single encrypted stream to support both layer access and resolution access rather than two differently encrypted codestreams used in [6]. No overhead data is provided in [6]. We expect that it has a similar overhead as our scheme when no truncation is applied (one seed in [6] and padding bits in ours for each code-block) but our scheme has less overhead if truncation is applied. In addition, our scheme has much less information leaking and much higher security than the schemes proposed in [6].

An alternative scheme is to use the ciphertext stealing method [8] in CBC mode which generates ciphertext of exactly the same size as the plaintext for any plaintext of size larger than one encryption block. In this method, the last full block and the last partial block of plaintext are encrypted differently from the rest blocks. In this alternative scheme, if a plaintext is not aligned with the encryption block size, the ciphertext of the last two blocks must be packed and truncated together in a packet. All other blocks are not allowed to be truncated to a partial encryption block. In this way, the last two blocks encrypted with ciphertext stealing can be detected if the ciphertext of a code-block inside a packet is not aligned with encryption block size. In a rare case that the compressed bitstream of a code-block is less than a full block, the ciphertext stealing method cannot be used. In this case, the padding scheme mentioned above is used. If this rare case is ignored, this alternative scheme does not introduce any overhead other than the inserted global IV.

The proposed J2K FGS encryption scheme is equally applicable to motion JPEG 2000. For motion J2K, an independent random "frame" IV is generated for each frame. The frame IV which plays the same role as the global IV in J2K encryption is then inserted into the header fields of the frame. Each frame is encrypted in the same way as the image case. Auxiliary data can be inserted into frame header fields to allow near RD-optimal truncation for each frame.

IV. Experimental Results

The proposed scheme has been implemented based on the publicly available J2K implementation JasPer (version 1.701.0) [9]. The block cipher and the hash function used in our implementation are Blowfish [8] and SHA-1 [8],

respectively, from the publicly available Crypto++ library (version 5.2.1) [10]. Blowfish is a 64 bit block cipher. The following reported results are based on the experiments on a set of standard 8-bit grayscale images of 512 by 512 pixels with the proposed scheme which does *not* use the ciphertext stealing method. Each image is compressed to a nominal 1.0 bpp with 5 levels of wavelet decomposition (i.e., 6 resolution levels) and 10 layers. The nominal code-block size is set to 64 by 64. Table I shows the overhead of the proposed encryption scheme when *no* truncation is applied. The overhead is about 1.0% for all those images, which is smaller than SSS reported in [7], even though our scheme has much finer granularity of scalability. We mentioned that the overhead of our scheme would reduce with truncations. This is confirmed in our experiments to be described next.

TABLE I. Encryption overhead for 512 by 512 grayscale images compressed at 1.0 bpp without any truncation.

Images	Barbara	Boat	Goldhill	Lena	Peppers	Zelda
Overhead (%)	1.00	0.91	1.02	0.96	0.95	0.95

Figure 1. Images of the three highest resolutions from the encrypted codestream of the image Lena.

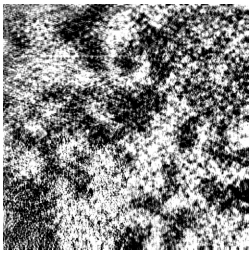

Figure 2. Encrypted Lena.

Figure 1 shows the images of the three highest resolutions supported with the encrypted codestream of Lena. Figure 2 shows the encrypted image Lena. It appears very random without any visual information leaking out. Figure 3 shows the RD curve truncated at each layer for the encrypted image Lena along with the original J2K RD curve without encryption. The two curves almost coincide with each other. This means that the RD performance when truncations are applied directly on an encrypted codestream produced by our scheme is almost the same as the original J2K coder for a large range of bitrates. Our scheme produces fine and smooth RD curves at each resolution, in great contrast to the stairstep-like RD curves when 3 resolutions are supported with SSS in the example given in [7]. Figure 3 also shows that our scheme's experimental points get closer to the corresponding points of non-encryption case when more layers are truncated, virtually overlapping each other at left side. This implies that the overhead of our scheme reduces when more data is truncated, and is negligible at low bitrates. SSS shows an opposite behavior that its overhead increases percentage-wise when more data is truncated.

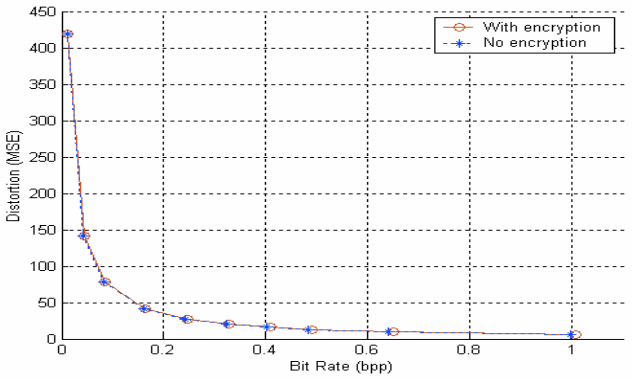

Figure 3. RD curves for both encrypted and unencrypted codestreams of the image Lena truncated at preset layers.

V. Conclusion

We have described a novel encryption scheme for JPEG 2000 which preserves in the encrypted codestream almost the same fine granularity scalability as the original unencrypted J2K codestream. The overhead is very small, about 1.0% for 8-bit 512 by 512 grayscale images at 1.0 bpp, when no truncation is applied. This overhead reduces to a negligible level when truncations are applied to code-blocks and more and more data is truncated. The described scheme produces encrypted codestreams with fine and smooth RD curves at each resolution for a large range of bitrates. We have also shown how the proposed scheme works with motion JPEG 2000.

References

[1] *Information Technology – JPEG 2000 Image Coding System, Part 1: Core Coding System*, ISO/IEC 15444-1:2000.

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

[3] R. Iannella, "Digital Rights Management (DRM) Architectures," *D-Lib Magazine*, vol. 7, no. 6, June 2001.

[4] A. M. Eskicioglu, J. Town, and E. J. Delp, "Security of Digital Entertainment Content from Creation to Consumption," *Signal Processing: Image Communication, Special Issue on Image Security*, vol. 18, no. 4, April 2003, pp. 237 – 262.

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

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

[7] 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. 14-17, 2003.

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

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

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