PARTIAL-SCRAMBLING OF IMAGES ENCODED USING JPEG2000 WITHOUT GENERATING MARKER CODES

Hitoshi KIYA, Shoko IMAIZUMI and Osamu WATANABE

Department of Electrical Engineering, Tokyo Metropolitan University,
1-1 Minamiosawa, Hachioji-shi, Tokyo 192-0397, Japan

ABSTRACT

A method is described for efficient partial-scrambling of JPEG2000 images that avoids generating marker codes and improves the ability to control the degree, strength, and computational complexity of scrambling. This higher control ability is due to the use of a parameter. This parameter also controls the scrambling time, an important consideration for real-time processing.

1. INTRODUCTION

The growing popularity of digital images has made the task of protecting copyrights and privacy more and more important because digital images can be easily duplicated and distributed. There are three main approaches to protecting copyrights and privacy: (a) cryptography (encrypting the whole image) [1], (b) digital watermarking [2], and (c) partial-scrambling [3, 4, 5, 6, 7]. This paper describes a new method for the third approach – partial-scrambling.

Cryptography eliminates any chance of evaluation or partial viewing by persons without the key, but is the most expensive approach. Digital watermarking is used less than cryptography and partial-scrambling because it does not degrade images virtually and cannot prevent unauthorized duplication. Moreover, it cannot protect privacy.

With partial-scrambling, images are encrypted and any images obtained are degraded to such an extent that they are barely recognizable. Viewers of an image can see its content and evaluate it to some degree without it being decrypted. Moreover, providers of images can distribute scrambled images using networks or storage devices, such as CD-ROMs. If a viewer has the decryption key and wants to view a de-scrambled image, he or she can de-scramble the image with a special decoder. In the standardization of JPEG2000 (Part8), partial-scrambling is described as a secure function [8, 9].

The scrambling of JPEG2000 images must meet at least the following requirements.

(a) Scrambling/de-scrambling should be applicable to JPEG2000 code-streams.

(b) Scrambling a code-stream should not change its size.

(c) A scrambled code-stream must be JPEG2000 Part-1 compliant.

(d) The scrambling method should be able to control the degree and strength of the scrambling.

(e) The scrambling method should be able to control the computational complexity of the scrambling.

Requirements(c), (d), and (e) are not met by conventional methods [4, 5, 6, 7]. The proposed method meets all of them.

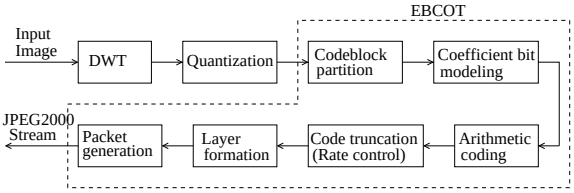


Fig. 1. JPEG2000 encoding

2. JPEG2000 CODING AND THE CONVENTIONAL SCRAMBLING METHODS

2.1. JPEG2000 coding [10, 11]

As illustrated in Fig. 1, JPEG2000 encoding can be divided into two parts – I) discrete wavelet transform and quantization and II) embedded block coding with optimized truncation (EBCOT) [12]. The input image is decomposed into sub-bands by discrete wavelet transform, and the wavelet coefficients of each sub-band are quantized. The quantized coefficients are coded using the EBCOT algorithm. The proposed method exploits the EBCOT algorithm. The EBCOT algorithm is divided into five parts: code-block partition, coefficient bit modeling, arithmetic coding and code truncation (rate control), layer formation, and packet generation.

An example structure of the JPEG2000 code-stream is shown in Fig. 2. It has a layer structure: a global header is followed by the most significant layer and so on to the least significant layer. The stream is terminated by a two-byte marker, the EOC (end of code-stream). The global header contains information necessary for decompressing the code-stream. Each layer is composed of a sequence of packets. Every packet in this packet-stream. Every packet consists of a header and a body and contains data divided into sub-bands except level 0. The unit of packets is determined by the resolution level. Level 0 contains the data of only LL, and the other levels contains three elements (HL, LH, HH).

In the conventional methods, layers, resolution levels, or code-blocks are selected as the unit for scrambling, and the body data is scrambled.

2.2. Requirements for scrambling

The proposed scrambling method should meet the following requirements [8, 9] which mentioned in section 1.

Figure 3 illustrates scrambling and de-scrambling with JPEG2000 coding. The JPEG2000 stream output by the encoder is decomposed into a header and a body by a parser, and only the body is

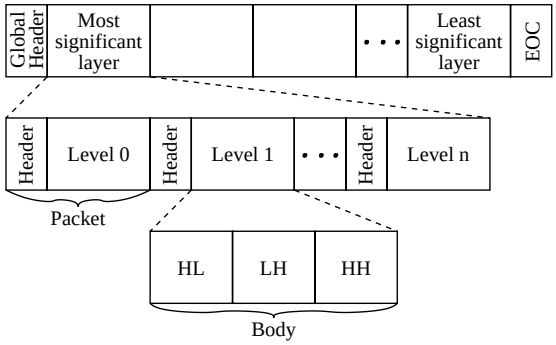
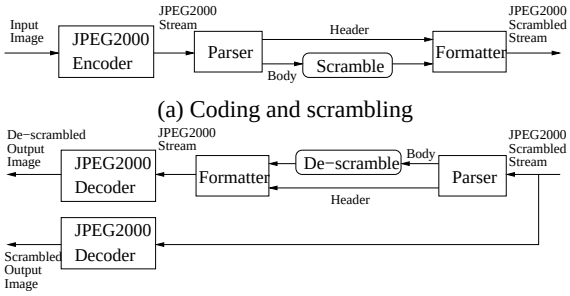


Fig. 2. JPEG2000 code-stream



(a) Coding and scrambling

(b) De-coding and de-scrambling

Fig. 3. Scrambling/de-scrambling with JPEG2000 codec

scrambled (requirement(a)). Scrambling does not affect the rate control and the header. Therefore, the amount of data is the same before and after scrambling (requirement(b)). A formatter reconstructs the stream as a scrambled stream. These steps are reversed for de-scrambling. A scrambled stream is de-scrambled, and the de-scrambled stream is sent to the decoder, which outputs the de-scrambled image (requirement(a)). However if the decoding side does not de-scramble and the scrambled stream is sent to the decoder, the output image is partially scrambled (requirement(c)).

The conventional methods [4, 5, 6, 7] scramble the body data, which directly represents images as layers, resolution levels, or code-blocks (requirement(a) and (b)). Selecting the layers, resolution levels, and code-blocks controls the degree and the strength of scrambling, i.e., the computational complexity of scrambling (requirement (d) and (e)). The conventional methods, however, must generate marker codes because they map the body data to random values using one byte in scrambling. Generation of marker codes does not meet requirement (c). Marker codes have a special meaning in JPEG2000. When a false marker code is generated by scrambling, the partially scrambled image cannot be decoded correctly. Hence, conventional methods do not meet requirement (c). Besides, the conventional methods do not completely meet requirement (d) and (e). The proposed method does meet requirement (c), (d), and (e), while retaining the advantages of the conventional methods.

2.3. Marker codes in JPEG2000 [11]

The marker codes in JPEG2000 are special command codes with values ranging from $FF90_h$ to $FFFF_h$, where "$_h$" means written in hexadecimal notations. All marker codes are represented with two bytes. The upper byte is FF_h, which is followed by another byte, xx_h. Therefore the marker codes are represented as $FFxx_h$.

The JPEG2000 standard assigns the codes to important delimiting code-stream markers. The provisions of the standard state that markers should never occur within the compressed data itself; that is, the arithmetic coder (MQ coder) should be designed not to generate these codes. We thus avoid using marker codes in the range $FF90_h$ to $FFFF_h$.

3. PROPOSED METHOD

As described above, in the conventional methods, layers, resolution levels or code-blocks are the units for scrambling, and the body data is scrambled using one byte. The proposed method scrambles the body data in a similar manner. The purpose of the proposed method is to eliminate the problems caused by the generation of marker codes in the conventional methods and to improve the ability to control the degree, strength, and computational complexity of scrambling.

3.1. Conditions for avoiding marker codes

Marker codes in the range $FF90_h$ to $FFFF_h$ are not normally generated in the encoded data itself. Thus, scrambling of the body data output by the MQ coder must not generate any marker codes in this range. Therefore, we propose the following conditions for avoiding marker codes.

(a1) If a byte is below 90_h, it must be kept being below 90_h after scrambling.

(a2) If a byte is 90_h or above (except FF_h), it must be kept being below FF_h after scrambling.

If the byte is FF_h, there is no scrambling restriction. The generation of marker codes can be completely avoided under these conditions.

Obviously, conventional methods, which map the body data to random values using one byte, cannot meet these conditions. To meet them, we use a half byte, not a byte, for scrambling. A byte is divided into an upper and a lower half byte. The conditions (a1) and (a2) are met, respectively, by conditions (b1) and (b2).

(b1) If a byte is below 90_h, the upper half byte must be kept being below 9_h after scrambling.

(b2) If a byte is 90_h or above (except FF_h), the upper or lower half byte must be kept below F_h after scrambling.

Several methods meet these conditions. We will describe one in detail.

3.2. Scrambling and de-scrambling

Here we describe a specific method that meets the conditions (b). The main differences from the conventional methods [4, 5, 6, 7] are that the generation of marker codes is avoided by using half-byte and that the ability to control the degree, strength, and computational complexity of scrambling is improved by using a parameter M.

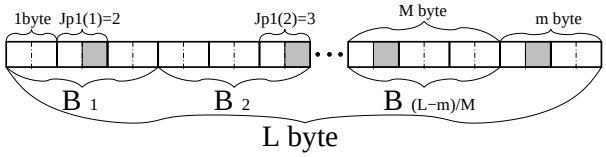


Fig. 4. Body data with L bytes ($M = 3, m = 2$); hatched parts are scrambled half bytes.

A. Scrambling

As shown in Fig. 3(a), the JPEG2000 stream output from the encoder is sent to the parser and decomposed into the header and body. Then the layers, resolution levels, and code-blocks are selected, and the body data (presupposed L bytes) is distilled. The body data is divided into blocks B_k ($k = 1, 2, ..., (L - m)/M, m = mod(L, M)$). Each block has M bytes. The following operation is done to each block B_k by byte(Fig. 4).

Step1. Using initial value p_1 and an algorithm for generating random integers $J_{p_1}(k)$, generate random integers in the range 1 to M:

$$J_{p_1}(k) \in \{1, 2, ..., M\}, k = 1, 2, ... \quad (1)$$

Step2. Selecting the $J_{p_1}(k)$-th byte from B_k individually.

Step3. Divide the selected $J_{p_1}(k)$-th byte into upper and lower half byte:
(3.1) If the selected byte is below $F0_h$, the lower half byte is selected.
(3.2) If the selected byte is $F0_h$ or above, the lower half byte or skip the byte (block B_k is excluded from scrambling) is selected.

Step4. Scramble only the half byte assigned in step3.

The scrambling must be not change the size of the code-stream. We describe two methods for doing this. One uses an encryption algorithm that maps one byte to another byte. In the Blowfish [13] encryption algorithms, for example, a byte is constructed of two lower half bytes selected as scrambling targets, and the byte is encrypted. If the selected byte is $F0_h$ or above in step (3.2), the byte must be skipped.

The other method uses the lower half byte of the selected byte as the scrambling target in step (3.2); that is, the lower half byte is selected as the scrambling target in both step (3.1) and (3.2). The bit-shift operation is done the selected lower half byte [9]. The lower half byte of the selected j-th one byte is

$$X_j = (x_1, x_2, x_3, x_4), x_i \in \{0, 1\}. \quad (2)$$

A left round shift of S_j bits,

$$S_j \in \{0, 1, 2, 3\}, \quad (3)$$

is performed to this X_j. Let $\hat{X}_j$ denote the transformed half byte. For example, when $S_j = 1$, $\hat{X}_j$ is written as

$$\hat{X}_j = (x_2, x_3, x_4, x_1). \quad (4)$$

The two methods described above does not generate marker codes and does not change the amount of data in the code-stream.

B. De-scrambling

The de-scrambling operation uses an algorithm for generating random integers ($J_{p_1}(k)$), a initial value of p_1, an area of L bytes for the body data (layers, resolution levels, and code-blocks), and a parameter M. As shown in Fig. 3(b), encoded data is sent to the parser and decomposed into the header and the body. The de-scrambling is done as follows.

Step1. With a initial value of p_1, random integers $J_{p_1}(k)$ are generated in the range 1 to M.

Step2. In each B_k, the $J_{p_1}(k)$-th byte selected on the scrambling side is located individually.

Step3. A half byte is selected to the located byte; it is the de-scrambling target.
(3.1) If the located byte is below $F0_h$, the lower half byte is selected.
(3.2) If the located byte is $F0_h$ or above, the lower half byte is selected or the byte is skipped (block B_k is excluded from de-scrambling).

Step4. Only the half byte selected in step3 is de-scrambled.

If the scrambling used encryption, de-scrambling requires the encryption algorithm and encryption keys. On the other hand, if the scrambling used bit-shift, the amount of the bit-shift is required. For the scrambling by bit-shift, right round shift of S_j bits is performed to $\hat{X}_j$ and then X_j is obtained.

As shown at the bottom of Fig. 3(b), if de-scrambling is not performed, the scrambled stream is sent to the decoder, and the partial-scrambled image is output.

4. EXPERIMENTS AND RESULTS

We evaluated the effectiveness of the proposed method by some experiments using the standard image "Lena" (256-level grayscale, 512×512). Image coding was done using five-level decomposition/composition based on two-channel filter banks with a Daubechies 9/7 bi-orthogonal wavelet filter running under the JPEG2000 verification model 8.7 [14]. The target bit rate was set to 1 bit/pixel, and scrambling was done using the bit-shift algorithm described in previous section.

A. Control of image quality

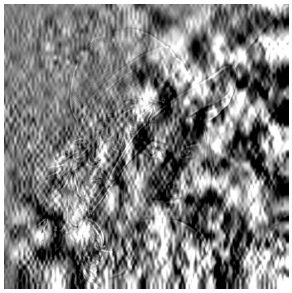
We show the images scrambled using JPEG2000 layer structure. Because the number of layers and the data size of each layer can be set comparatively freely using this function, it is easy to control the degree and computational complexity of the scrambling. In the image encoding, 50 layers were formed (layer 0 to layer 49), and each layer shared 0.02 bit/pixel. Figure 5(a) shows the scrambled image when layer 2 (the most significant layer is layer 0) was selected as scrambling target. Because M was set to 1, all of the data in the selected layer was scrambled, as in the conventional methods; that is, all the encoded data was decoded. The image quality was the same as with conventional methods, but generation of marker codes was avoided. If layers upper than layer 2 are selected, the degree of scrambling can be strengthened.

Figure 5(b) shows a scrambled image when only layers 0 and 1 were decoded, the other layers were dropped. The image quality is better than in Fig. 5(a). If the upper layers that are not scrambled are decoded, the image quality is improved and the effect of scrambling is easily reduced. To avoid this problem, the conventional methods must select resolution levels that have poor controllability, instead of layers, as the scrambled area.

(a) decoding all layers

(b) decoding only Layers 0 and 1

Fig. 5. The example of proposed scramble using JPEG2000 layer structure (scrambling position: Layer 2, $M = 1$)



(a) $M = 1$

(b) $M = 128$

Fig. 6. The example of proposed scramble using SNR scalability function and parameter M (scrambling position: Layer 0)

Figure 6 shows a scrambled image when layer 0 was selected as the target of scrambling. The selection of layer 0 prevents improvement in image quality by dropping the lower layers. However, conventional methods ($M = 1$) generate invisible images when layer 0 is selected. The proposed method can generate partial-scrambled images by adjusting the value of M even if layer 0 is selected, as shown in Fig. 6(b).

B. Control of scrambling time

Parameter M controls not only the image-quality(i.e., the strength of scramble) but also scrambling time. The proposed method is thus suitable for applications demanding real-time processing.

5. CONCLUSION

We have described a new method for partial-scrambling of images based on JPEG2000. It avoids the generation of marker codes and decodes images correctly. It works by scrambling the body data using a half byte with some restrictions. It better controls the degree, strength, and computational complexity of scrambling. By providing some experiments and those results, it has been confirmed that the proposed method is effective for partial-scrambling of JPEG2000 images.

6. ACKNOWLEDGEMENT

The authors wish to thank Mr. Takahiro Fukuhara, Mr. Seiji Kimura and Mr. Katsutoshi Ando of Sony Corporation for their useful advice.

7. REFERENCES

[1] H. Kinoshita, N. Shioiri and Y. Sakai,“Appropriate Image Data Encipherment Method for DCT Coding,” IEICE Trans., Vol. J75-D-I, No. 5, pp. 314–321, May. 1992.

[2] G.C. Langelaar, I. Setyawan, and R.L. Lagendijk, “Watermarking Digital Image and Video Data,” IEEE Signal Processing Magazine, Vol. 17, No. 5, pp. 20–46, Sep. 2000.

[3] H. Fujii and Y. Yamanaka, “Digital Image Scrambling Method for Information Distribution,” IPSJ Trans., Vol. 38, No. 10, pp. 1945-1955, Oct. 1997.

[4] R. Grosbois, P. Gerbelot and T. Ebrahimi, “Authentication and access control in the JPEG2000 compressed domain,” Proc. SPIE 46th Annual Meeting, Applications of Digital Image Processing XXIV, Vol. 4472, pp. 95–104, San Diego, July 29th-August 3rd, 2001.

[5] K. Ando, O. Watanabe and H. Kiya, “Partial-scrambling of still images based on JPEG2000,” International Conference on Information, Communications, and Signal Processing,2F2.5, CD-ROM, Singapore, Oct. 2001.

[6] K. Ando, O. Watanabe and H. Kiya, “Partial-scrambling of Images Encoded by JPEG2000,” IEICE Trans., Vol. J85-D-II, No. 2, pp. 282–290, Feb. 2002.

[7] K. Ando and H. Kiya, “An Encryption Method for JPEG2000 Images Using Layer Function,” IEICE Trans., Vol. J85-A, No. 10, pp. 1091–1099, Oct. 2002.

[8] T. Fukuhara, K. Ando, O. Watanabe and H. Kiya, “Partial-scrambling of JPEG2000 Images for Security Applications”, ISO/IEC JTC 1/SC29/WG1, N2430.

[9] T. Fukuhara, K. Ando and H. Kiya, “Proposal of conditional access and signalling of security parameter for JPSEC”, ISO/IEC JTC 1/SC29/WG1, N2721.

[10] ISO/IEC IS 15444-1: “Information technology - JPEG2000 image coding system - Part 1: Core coding system,” 2000.

[11] D. Taubman and M. Marcellin, JPEG2000 Image Compression Fundamentals, Standard and Practice, Kluwer Academic Publishers, Jan. 2002.

[12] D. Taubman, “High Performance Scalable Image Compression with EBCOT,” IEEE Trans. Image Processing, Vol. 9, No. 7, pp. 1158–1170, Jul. 2000.

[13] B. Schneier, “Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish),” Fast Software Encryption, Cambridge Security Workshop Proceedings (Dec. 1993), Springer-Verlag, 1994, pp. 191-204.

[14] ISO/IEC JTC 1/SC29/WG1 N1894: “JPEG2000 Verification Model 8.6 Software,” 2000.