Bitstream-Based JPEG Image Encryption with File-Size Preserving

Hiroyuki KOBAYASHI
Tokyo Metropolitan College
of Industrial Technology,
Tokyo, JAPAN
Email: hkob@metro-cit.ac.jp

Hitoshi KIYA
Tokyo Metropolitan University
Tokyo, JAPAN
Email: kiya@tmu.ac.jp

***Abstract*—An encryption scheme of JPEG images in the bitstream domain is proposed. The proposed scheme preserves the JPEG format even after encrypting the images, and the file size of encrypted images is the exact same as that of the original JPEG images. Several methods for encrypting JPEG images in the bitstream domain have been proposed. However, since some marker codes are generated or lost in the encryption process, the file size of JPEG bitstreams is generally changed due to the encryption operations. The proposed method inputs JPEG bitstreams and selectively encrypts the additional bit components of the Huffman code in the bitstreams. This feature allows us to have encrypted images with the same data size as that recoded in the image transmission process, when JPEG images are replaced with the encrypted ones by the hooking, so that the image transmission are successfully carried out after the hooking.**

***Index Terms*—JPEG, Encryption, File-size preserving, Bitstream-based**

I. Introduction

Due to the spread of digital cameras and smart phones, opportunities to use digital images are increasing. Generally, captured images are immediately JPEG encoded and stored. These images are not only stored on personal devices, but also are often uploaded to cloud providers, such as social networks, cloud photo storage services, and so on. Most the cloud providers accept only limited file formats like JPEG. In addition, such cloud environments are based on the reliability of the providers, but they are not a reliable situation in terms of privacy preserving for users. Therefore, various image encrypting methods have been studied for compressed image data.

Encryption then Compression(EtC) systems [1]–[11] are methods to encrypt for images before encoding. Some of these methods have the compatibility with international compression standards, and enable privacy preserving decompression and compression.The compression performances of EtC systems are almost as same as one of the original images, but the file sizes are slightly different from those of compressed images without encryption.

Several bitstream based encryption methods have also been proposed [12]–[17]. For JPEG 2000 images, some bitstream-based encryption methods have been proposed in consideration of the generation of special marker codes and without changing the file size [12]–[14]. Even for JPEG images, some bitstream-based encryption methods have been proposed, but the file size of the encrypted bitstream has changed by the occurrences or disappearances of the JPEG marker code [15]–[17]. For example, in [15], encryption is performed while keeping the file size by rearranging the run lengths of AC coefficients. However, occurrences or disappearances of the pseudo marker code have not been studied. In this case, the encrypted bitstream can not be correctly decoded or the file size of the encrypted bitstream is changed. On the other hand, in the methods of [16], [17], bitstream-based block scrambling and coefficient scrambling are implemented. In these methods, in order to accurately hold the JPEG format, encryption processing is executed the byte stuffing operation which prevent accidental generation of markers by the arithmetic encoding procedures. As a result, It has been shown that the file size changes by several bytes before and after encryption.

In this paper, we propose a bitstream-based JPEG encryption scheme that makes the file size exactly equal to the original. The proposed method guarantees a constant file size by providing a mechanism to avoid occurrences / disappearances of the JPEG marker code. This feature allows us to have encrypted images with the same data size as that stored in the image transmission process, when JPEG images are replaced with the encrypted ones by the hooking, so that the image transmission are successfully carried out after the hooking.

II. JPEG Bitstream and Its Byte Stuffing

A. JPEG Bitstream

Figure 1(a) shows the structure of a JPEG bitstream. SOI and EOI are the marker codes which correspond to “Start of Image” and “End of Image”, respectively. JPEG bitstreams have some marker segments (“Segment” in Fig.1(a)) which store information to be used for data decoding such as quantization tables, Huffman tables, and so on. Each marker segment starts with a marker code. The marker codes are special two-byte codes where the first byte is “FF” and the second byte is a value between “01” and “FE”.

Figure 1(b) shows the structure of the “image data” in Fig.1(a). “Image data” consists of multiple MCUs (Minimum Coded Unit). Figure 1(b) is an example in the case of 4:2:0 color subsampling. Therefore, each MCU has four Y(luminance) blocks, one subsampled Cb block, and one

978-1-5386-6309-7/18/$31.00 ©2018 IEEE

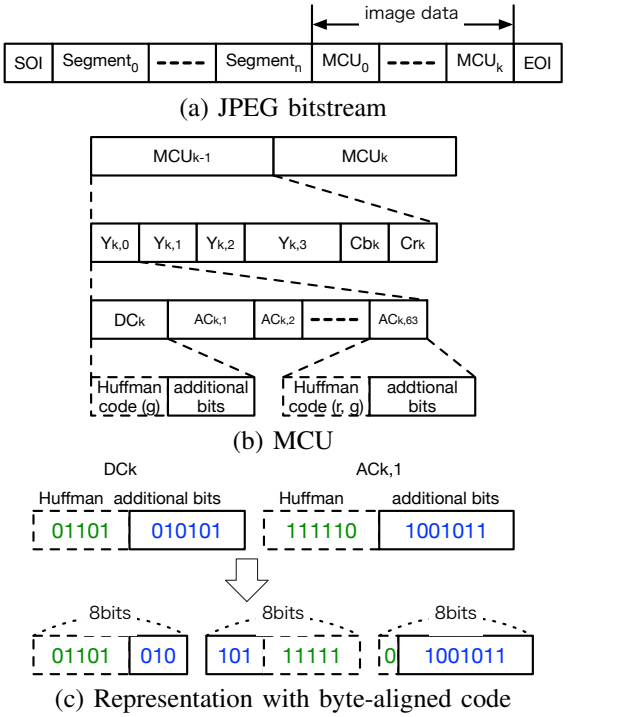


Fig. 1. Structure of a JPEG bitstream

subsampled Cr block. Moreover, each block has DC_k which is the difference value from the DC coefficient of previous block, and 63 AC coefficients $AC_{k,n}$ $(n = 1 \cdots 63)$. Each coefficient has a Huffman code part corresponding to the group number (g) that determines the range of values, and an additional bits part for uniquely identifying values within the range. In the case of AC coefficients, the Huffman code also includes the run length (r) of the zero value until a significant value exists.

Figure 1 (c) is an example of Huffman code and additional bits. Since each data is composed of variable length bits, they are stored in byte units.

B. Difficulty of file size preserving

Figure 2 illustrates an example of entropy-coded data segment, where DC_k in Fig.2(a) is entropy-coded DC coefficient data and $AC_{k,1}$ is the entropy-coded data of the first AC component in the k-th block, respectively. Each data have Huffman code part and additional bits part. To generate the JPEG bitstream, byte-based packing is first applied to the entropy-coded data in Fig.2(a), as shown in Fig.2(b). The byte "FF", i.e. "11111111", which corresponds to a marker code, may be produced due to the byte-based packing. Therefore, finally, in order to ensure that the marker does not occur within an entropy-coded segment, any "FF" byte in either a Huffman or additional bits, is followed by a "stuffed" zero byte, whose operation is called as 'byte stuffing', as shown in Fig.2(c) [18].

Note that the file size of the stream (b), is not the same as that of the stream (c). We have to consider the byte stuffing operation to preserve the same file size when JPEG images are encrypted.

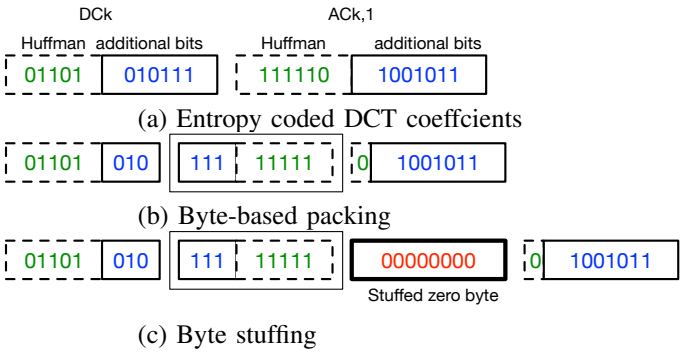


Fig. 2. Byte stuffing in entropy-coded data segment

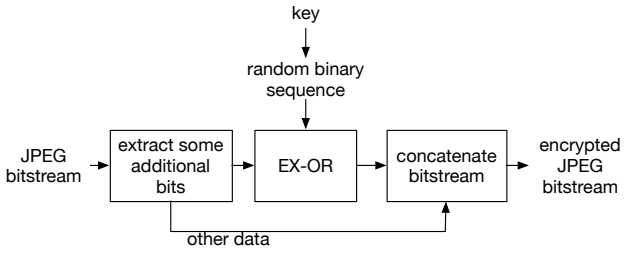


Fig. 3. Outline of the proposed method

III. Proposed Structure

We propose a new bitstream-based JPEG image encryption method which allows us to exactly preserve the same file size as the original JPEG bitstream.

A. Outline of the proposed structure

Bitstreams encrypted by the proposed method have not only the same file sizes, but also the compatibility with JPEG decoders. Some of only additional bits fields that satisfy conditions are encrypted to keep the compatibility with JPEG decoders. Figure 3 illustrates the outline of the proposed method. The procedure of the proposed method is summarized as follows.

1) Analysis, byte-by-byte, the entropy-coded data segment and extract additional bits from a byte that satisfies two conditions: the byte includes both Huffman code and additional bits, and the Huffman code includes at least one "0" bit.
2) Generate a random binary sequence with a secret key.
3) Carry out exclusive-or operation between only extracted additional bits and the random sequence generated in 2), and replace the additional bits with the result.
4) Produce an encrypted bitstream by combining the encrypted additional bits with other data without any encryption.

B. Encryption considering occurrences or disappearances of 'FF00'

If all additional bits are simply encrypted, there is a possibility to generate or lose "FF". Therefore, in the proposed method, only limited additional bits are encrypted based on exclusive-or operation with a random binary sequence.

Fig. 4. Example for determination whether encrypt or not

In Fig.4, the additional bits in Fig.2(c) were replaced with 'x'. Using Fig.4, we describe an outline of determination whether encryption is possible or not.

1) The first byte: The data consist of a 5-bit Huffman code and 3-bit additional bits. Even if all the additional bits are 1, the entire byte never becomes "FF". Therefore, the additional bits part is able to be encrypted.

2) The second byte: The data consist of 3-bit additional bits and a 5-bit Huffman code. In the original data, since the additional bit was "111" and the remaining Huffman code was '111111", "FF" was composed as the whole byte. If any bit of the additional bit is changed to 0 by encryption, the entire byte is not 'FF' and '00' of the third byte is not inserted. Since this causes a file size change, the additional bits of the second byte are not encrypted.

3) The third byte: The data are padding data because the second byte is "FF". Since this is not an additional bit, it is not encrypted.

4) The last byte: The data consist of a 1-bit Huffman code and 7-bit additional bits. Even if all the additional bits are 1, the entire byte never becomes "FF". Therefore, the additional bits part is able to be encrypted.

By analyzing the Huffman code in the byte as described above, it is possible to determine whether encryption is possible or not. In summary, the following bytes are not encrypted.

1) The whole 8 bits are Huffman codes.
2) The whole 8 bits are additional bits.
3) "00" byte immediately after "FF".
4) Huffman code and additional bits are included and the all bits of Huffman code are '1'.

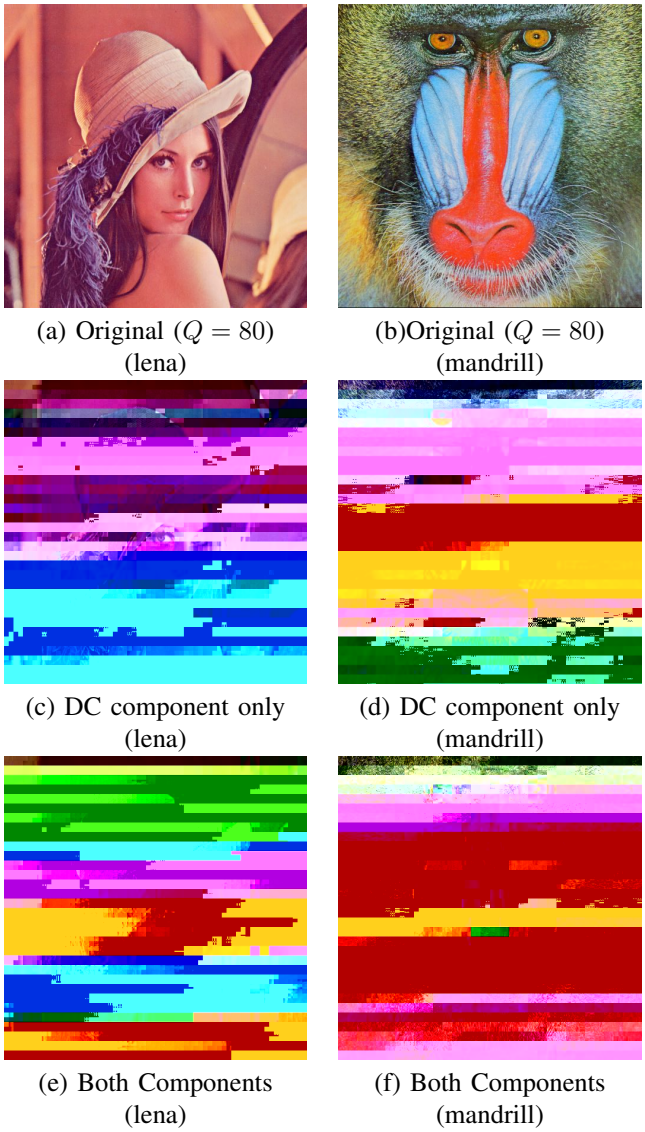
IV. Experimental Results and Discussion

Some simulations were carried out to demonstrate the effectiveness of the proposed method. For the simulations, we used the reference software distributed by JPEG [19] with 4:2:0 chroma subsampling.

A. Image quality evaluation of encrypted image

First, the image quality of encrypted images was evaluated. Figure 5(a) and (b) were standard images, "lena" and "mandrill" encoded by JPEG with Q-factor 80, respectively. Figure 5(c) to (f) were decoded images with a standard JPEG decoder from the encrypted images by the proposed scheme. In Fig.5(c) and (d), the additional bits in only DC components were encrypted. On the other hand, in Fig.5(e) and (f), the additional bits in both DC and AC components were encrypted.

The encrypted image in Fig.5(b) has slightly visible information on the original one, because the AC components were the same as the original ones. On the other hand, the encrypted image in Fig.5(c) had less visible information than Fig.5(b).



(a) Original ($Q = 80$) (lena)
(b)Original ($Q = 80$) (mandrill)
(c) DC component only (lena)
(d) DC component only (mandrill)
(e) Both Components (lena)
(f) Both Components (mandrill)

Fig. 5. Original and encrypted results (lena, mandrill)

B. The number of bytes to be encrypted

Table I indicates the ratio of encryption applied to additional bits in "image data". Data excluded from encryption are data which satisfy conditions 2) and 4) of **III-B**. Due to the increase of Q-factors, the proportion of encryption targets decreased. This is because the number of bytes corresponding to the condition 2) necessity increases due to the increase of Q-factors.

C. File-size preserving

Next, we compare our method with the previous works [1], [17], in terms of the file sizes. Table II shows the file sizes of encrypted JPEG images under various conditions. From this table, JPEG images encrypted by the proposed method had exactly the same file sizes as those of the original ones. However, other encryption methods could not preserve the same file sizes, because they do not consider the effect of byte stuffing.

TABLE I
THE NUMBER OF BYTES TO ENCRYPT / BYTES TO BE EXCLUDED FROM ENCRYPTION (LENA IMAGE)

(a) $Q = 50$

target	# of bytes excluded from encryption[byte]	# of encrypted bytes[byte]	Percentage of encrypted bytes[%]
DC only	70	4,729	98.5
AC only	172	9,591	98.2
Both	197	13,467	98.6

(b) $Q = 80$

target	# of bytes excluded from encryption[byte]	# of encrypted bytes[byte]	Percentage of encrypted bytes[%]
DC only	420	6,306	93.8
AC only	460	20,931	97.8
Both	741	26,104	97.2

(c) $Q = 95$

target	# of bytes excluded from encryption[byte]	# of encrypted bytes[byte]	Percentage of encrypted bytes[%]
DC only	1,758	7,152	80.2
AC only	3,225	59,063	94.8
Both	4,336	65,274	93.8

TABLE II
LENGTH OF ORIGINAL AND ENCRYPTED IMAGES (DIFFERENCE)[BYTE], FOR LENA IMAGE

Q-factor	50	80	95
Original	24,279	43,879	106,548
Proposed	**24,279(0)**	**43,879(0)**	**106,548(0)**
Cheng [17]	24,281(+2)	43,865(-14)	106,553(+5)
EtC [1]	24,767(+488)	44,487(+608)	108,262(+1,714)

V. CONCLUSION

In this paper, we have proposed an encryption method that allows us to preserve the same file size before and after the encryption. In the encryption process, the proposed method considers the effect of byte stuffing and guarantees a constant file size by providing a mechanism to avoid the occurrence or disappearance of the JPEG marker code. By preserving the file size, it can be expected that the image transmission are successfully carried out after the hooking encryption process.

REFERENCES

[1] Kenta Kurihara, Masanori Kikuchi, Shoko Imaizumi, Sayaka Shiota, and Hitoshi Kiya: "An Encryption-then-Compression System for JPEG / Motion JPEG Standard," IEICE Trans. Fundamentals, vol.E98-A, no.11, pp.2238–2245, November 2015.

[6] K. Kurihara, O. Watanabe, and H. Kiya, "An encryption-then- compression system for jpeg XR standard," in IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), 2016, pp. 1-5.

[2] J. Zhou, X. Liu, O. C. Au, and Y. Y. Tang, "Designing an efficient image encryption-then-compression system via prediction error clustering and random permutation," IEEE Trans. on information forensics and security, vol. 9, no. 1, pp. 39-50, 2014.

[3] O. Watanabe, A. Uchida, T. Fukuhara, and H. Kiya, "An encryption-then-compression system for JPEG 2000 standard," IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2015, pp. 1226-1230.

[4] K. Kurihara, S. Shiota, and H. Kiya, "An encryption-then-compression system for jpeg standard," Picture Coding Symposium (PCS), 2015, pp. 119-123.

[5] K. Kurihara, M. Kikuchi, S. Imaizumi, S. Shiota, and H. Kiya, "An Encryption-then-Compression System for JPEG / Motion JPEG Standard," IEICE Trans. Fundamentals, vol.E98-A, no.11, pp.2238-2245, November 2015.

[7] K. Kurihara, S. Imaizumi, S. Shiota, and H. Kiya, "An encryption-then-compression system for lossless image compression standards," IEICE Trans. Inf. & Syst., vol. E100-D, no. 1, pp. 52-56, 2017.

[8] T. Chuman, K. Kurihara, and H. Kiya, "On the security of block scrambling-based etc systems against jigsaw puzzle solver attacks," IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017, pp. 2157-2161.

[9] T. Chuman, K. Kurihara, and H. Kiya, "Security evaluation for block scrambling-based etc systems against extended jigsaw puzzle solver attacks," IEEE International Conference on Multimedia and Expo (ICME), 2017, pp. 229-234.

[10] Tatsuya Chuman, Kenta Iida, and Hitoshi Kiya, "Image Manipulation on Social Media for Encryption-then-Compression Systems," Proc. APSIPA Annual Summit and Conference, Kuala Lumpur, Malaysia, 14th December, 2017.

[11] Tatsuya Chuman, Kenta Kurihara, and Hitoshi Kiya "On the Security of Block Scrambling-based EtC Systems against Extended Jigsaw Puzzle Solver Attacks," IEICE Trans. Inf. & Sys., vol.E101-D, no.1, pp.37-44, January 2018.

[12] Ikeda, H., Iwamura, K.: Selective encryption scheme and mode to avoid generating marker codes in JPEG2000 code streams with block cipher. In: Proceedings of IEEE WAINA, pp. 593–600 (2011)

[13] H. Kiya, S. Imaizumi, and O. Watanabe, "Partial-Scrambling of Image Encoded Using JPEG2000 without Generating Marker Codes," Proc. IEEE International Conference on Image Processing(ICIP), no.WA-P1.3, 17th September, 2003.

[14] Hitoshi Kiya, Shoko Imaizumi, and Osamu Watanabe, "Partial-Scrambling of Image Encoded Using JPEG2000 without Generating Marker Codes," Proc. IEEE International Conference on Image Processing, no.WA-P1.3, Barcelona, Spain, 17th September, 2003.

[15] Unterweger, Andreas and Uhl, Andreas: "Length-preserving Bit-stream-based JPEG Encryption, " Proceedings of the on Multimedia and Security, MM&Sec '12, pp. 85–90, 2012.

[16] X. Niu, C. Zhou, J. Ding and B. Yang, "JPEG Encryption with File Size Preservation," 2008 International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Harbin, 2008, pp. 308-311.

[17] Cheng, Hang and Zhang, Xinpeng and Yu, Jiang and Zhang, Yuan: "Encrypted JPEG Image Retrieval Using Block-wise Feature Comparison, " J. Vis. Comun. Image Represent., Vol.40, pp. 111–117, 2016.

[18] "Information technology–Digital compression and coding of continuous-tone still images: Requirements and guidelines," International Standard ISO/IEC IS-10918-1, Feb. 1994.

[19] "Text of CD ISO/IEC 18477-5 (Reference Software)," ISO/IEC JTC 1/SC 29/WG 1 N69019, Jun. 2015.