




Article

Encryption Method for JPEG Bitstreams for Partially Disclosing Visual Information

Mare Hirose ¹, Shoko Imaizumi ^{2,*} and Hitoshi Kiya ^{3,*}¹ Faculty of Engineering, Chiba University, 1-33 Yayoicho, Chiba 263-8522, Japan; marehirose@chiba-u.jp² Graduate School of Engineering, Chiba University, 1-33 Yayoicho, Chiba 263-8522, Japan³ Faculty of System Design, Tokyo Metropolitan University, 6-6 Asahigaoka, Tokyo 191-0065, Japan

* Correspondence: imaizumi@chiba-u.jp (S.I.); kiya@tmu.ac.jp (H.K.)

Abstract: In this paper, we propose a novel encryption method for JPEG bitstreams in which encrypted data can preserve the JPEG file format with the same size as that without encryption. Accordingly, data encrypted with the method can be decoded without any modification of header information by using a standard JPEG decoder. In addition, the method makes two contributions that conventional methods allowing bitstream-level encryption do not: spatially partial encryption and block-permutation-based encryption. To achieve this, we propose using a code called restart marker for the first time, which can be inserted at regular intervals between minimum coded units (MCUs) for encryption. This allows us to define extended blocks separated by restart markers, so the two contributions are possible with restart markers. In experiments, the effectiveness of the method is verified in terms of file size preservation and the visibility of encrypted images.

Keywords: JPEG bitstream; partial encryption; restart marker; file size preserving



Citation: Hirose, M.; Imaizumi, S.; Kiya, H. Encryption Method for JPEG Bitstreams for Partially Disclosing Visual Information. *Electronics* **2024**, *13*, 2016. <https://doi.org/10.3390/electronics13112016>

Academic Editors: Zhenghao Shi, Miaohua Zhang, Feng Zhao, Lifeng He and Jihua Zhu

Received: 15 March 2024

Revised: 16 May 2024

Accepted: 20 May 2024

Published: 22 May 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The use of JPEG images has greatly increased because of the rapid growth of the Internet and widespread use of cloud-based services. Many of these services do not allow the use of arbitrary file formats and require compliance with the JPEG standard. In addition, cloud services are generally unreliable due to incidents such as information leakage. Accordingly, various encryption methods that can preserve the JPEG format have been studied so far [1–17]. An example of encryption methods that can maintain the JPEG format is methods used in encryption-then-compression (EtC) systems [1–3], although the traditional way of securely transmitting images is to use a compression-then-encryption (CtE) system. In EtC systems, image encryption is carried out prior to image compression. Therefore, these encryption methods cannot be applied to JPEG bitstreams. In addition, the file size of encrypted images is not the same as that of JPEG images without encryption. When an image file is hooked in a transmission channel, the file size information is transferred between applications and servers [16–18]. In such a case, the file size of encrypted JPEG images should be equal to that before encryption.

Various bitstream-level encryption methods for JPEG compression have also been studied to directly encrypt JPEG images [4–9,16,17]. Niu et al. [4] proposed a method that first encrypts only DC coefficients and then scrambles the positions of DCT blocks and embeds the information into AC coefficients. In Cheng et al.'s method [5], the positions of DC and AC coefficients in each DCT block are permuted, and then the DC coefficients and the quantization table are encrypted. In He et al.'s method [6], the substitution of DC coefficients considering overflows and the substitution of MCUs not containing DC coefficients guarantee that the encrypted image is compatible with the JPEG standard. In Qin et al.'s method [7], the error histogram of an adaptive DC coefficient prediction is encrypted. The AC coefficient-run-length pairs in the low-frequency domain are swapped throughout the image, and the number of AC coefficients is manipulated so that it is less

than 63 to prevent overflow. Yual et al. [8] proposed two methods that permute the AC coefficients among different blocks in order to enhance the security. One is used to change the number of non-zero coefficients and the energy of the AC coefficients, and the other is used to change the position of the last non-zero coefficients.

Image data encrypted with these methods can preserve the JPEG format. However, the file size of encrypted JPEG images is not equal to that of JPEG images without encryption. In addition, the conventional methods cannot generate partially encrypted images that have encrypted and unencrypted regions in an encrypted image.

There are other methods that can generate encrypted JPEG images [10–15]. In the methods proposed in [10–12], encryption is applied in the DCT domain after quantization, and a format-compliant encoded sequence is output. However, these methods cannot also preserve the file size of original JPEG images without encryption, and they cannot generate partially encrypted images.

Kobayashi et al. proposed a bitstream-based JPEG encryption method in which the file size is unchanged before and after encryption [16,17]. Furthermore, encryption can be performed independently of the application, e.g., by an https proxy server. However, this method cannot generate partially encrypted images. In addition, since the positions of DCT coefficients cannot be permuted, the visibility of encrypted images is not low enough, so we can sometimes recognize the visual information of original images partly from encrypted images.

Accordingly, we propose a novel encryption method for JPEG bitstreams by expanding the method [16,17]. The proposed method allows us to generate partially encrypted images while maintaining the same file size as that without encryption. We use a code called restart marker, which can be inserted at regular intervals between MCUs. The restart marker is a standard function described in the ITU, and it has been used for error propagation [19], but here, the marker is applied to encryption for the first time. This allows us to define extended blocks separated by restart markers, so the random permutation of extended blocks can be used for image encryption so that the visibility of encrypted images is variable.

Our contribution is to propose a novel method that allows us not only to obtain encrypted images with the same file sizes as images without encryption but to also choose the visibility of encrypted images from various modes. In experiments, the effectiveness of the method was verified in terms of the generation of partially encrypted images, the visibility of encrypted images, and the file size of encrypted images.

The rest of this paper is structured as follows. Section 2 presents the preparation for JPEG bitstreams and the method proposed in [16,17]. Section 3 puts forward the proposed method. Experiments and results are presented in Section 4. Section 5 concludes this paper.

2. Related Work

We aim to propose a novel encryption method combined with image compression that can generate JPEG files with hidden visual information. Accordingly, previous encryption methods combined with image compression are summarized here. In addition, the structure of JPEG bitstreams and the encryption method for JPEG bitstreams presented in [16,17] are briefly explained to clearly indicate our contributions.

2.1. Combined Use of Encryption with Image Compression

Encryption methods combined with image compression are classified into the four types below.

- Type 1: Compression -then-encryption with standard cryptography.
For Type 1, images are compressed and then encrypted with a standard cryptography such as AES (advanced encryption standard) (see Figure 1a) [20–22]. However, the format of the encrypted data is different from that of compressed data, such as in the case of the JPEG format, so the data cannot be uploaded to most cloud services such as Google Photos.
- Type 2: Encryption-then-compression.

For Type 2, the visual information of images is protected by using a perceptual encryption method, called a compressible encryption method, and then the encrypted images are compressed with a compression method [1–3,23,24] (see Figure 1b). When using the JPEG compression method, the compressed data can maintain the JPEG format [1–3]. In addition, this type of encryption has various applications including privacy-preserving deep learning [25,26]. However, given JPEG images prior to encryption, the JPEG images must be decompressed before carrying out encryption. This restriction causes the quality of images to degrade in addition to incurring additional computational costs.

Type 3: Joint encryption and compression.

For Type 3, encryption and compression are simultaneously carried out (see Figure 1c) [10–15]. For example, the value and position of DCT coefficients are randomly permuted to generate encrypted JPEG images in the middle of encoding. Thus, given JPEG images prior to encryption, the JPEG images must be decompressed before carrying out encryption as well. In addition, a non-standard encoder has to be prepared.

Type 4: Compression-then-encryption with a bitstream-level encryption method.

For Type 4, images are compressed and then encrypted with a bitstream-level encryption method [4–9,16,17], where bitstream-level encryption is carried out with a stream cipher in many cases (see Figure 1d). Type 4 encryption allows us not only to maintain the JPEG format but to also generate encrypted JPEG images with the same size as that of JPEG images without encryption under some requirements. Type 4 encryption also does not require any modification of encoders and decoders. Encrypted data can be decoded by using a standard JPEG decoder. In addition, the feature of the file size not changing makes it possible to implement an encryption system using mitmproxy [27,28], which is an open-source interactive HTTPS proxy, on a proxy server [17].

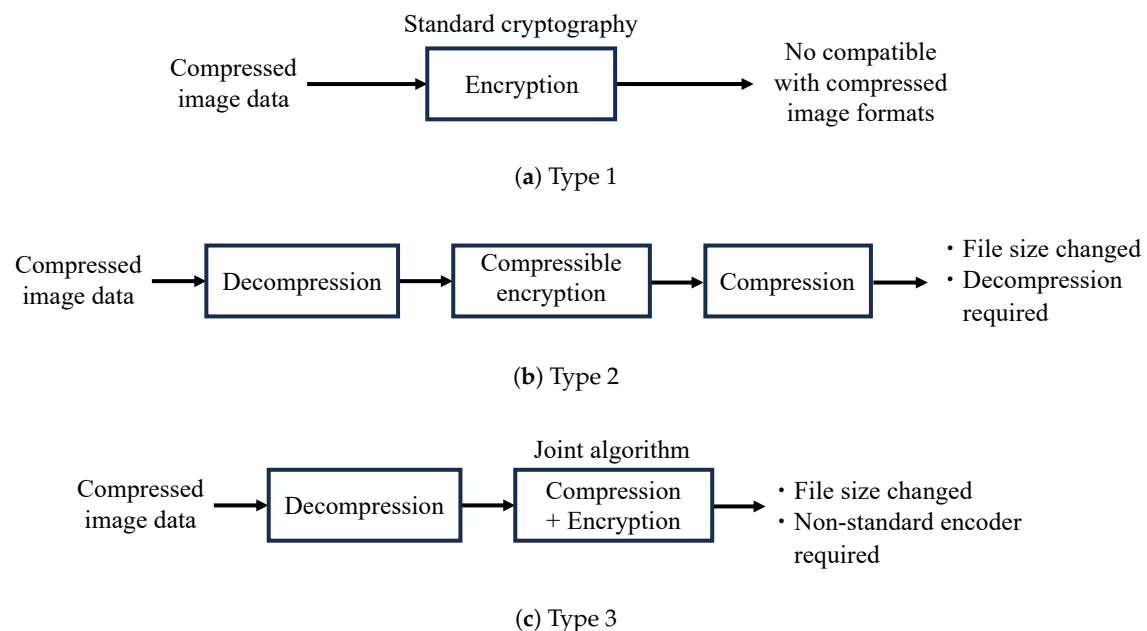


Figure 1. Cont.

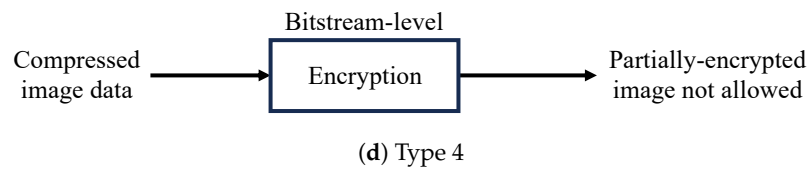


Figure 1. Combined use of encryption and compression.

We focus on Type 4 encryption for JPEG images in this paper. Various Type 4 encryption methods for JPEG images have been studied so far. This type has superior properties, compared with other types, but it cannot generate images that have yet to be encrypted with various visibility modes, including partially encrypted images. Increased freedom of visibility is important for improving the convenience of encrypted images and expanding applications.

2.2. JPEG Bitstreams

The proposed method handles the information of DC and AC coefficients in a bitstream, so the structure of JPEG bitstreams is reviewed here. Figure 2 shows an example of the structure of JPEG bitstreams [29], where SOI and EOI indicate the start of an image and the end of an image, respectively. The area between them, which consists of multiple segments and image data, is called a frame.

SOI and EOI are special codes with two bytes, called marker code. The first byte of any marker code is fixed to $FF_{(16)}$, i.e., “11111111”. Segments in a JPEG bitstream also have $FF_{(16)}$ in the first byte and contain information used for decoding, such as a Huffman table and a quantization table, in the second byte. Each segment can be identified by using the second byte of the code. Note that $00_{(16)}$ is outside the definition of marker codes.



Figure 2. Structure of JPEG bitstream.

Figure 3 shows the construction of image data in a JPEG bitstream. Image data are divided into MCU units, and each MCU unit consists of four luminance components (Y) and two chroma components (C_b , C_r), where the figure shows an example of 4:2:0 color subsampling. Information of the DC and AC coefficients is included in each component as Huffman codes and additional bits, although DC coefficients are stored as the difference value from the previous DC coefficients. These bit sequences are stored in byte units, so byte $FF_{(16)}$ occurs. In these cases, JPEG encoders insert $00_{(16)}$ just after $FF_{(16)}$ in image data so as to distinguish this $FF_{(16)}$ from the marker code. Moreover, JPEG decoders read only $FF_{(16)}$ and skips $00_{(16)}$ when $FF00_{(16)}$ is detected. This operation is called “byte stuffing”.

Restart marker (RST) is one of the marker codes. It can be placed at certain intervals between MCUs. If an error occurs in a DC coefficient in a bitstream, the error propagates and affects other DC coefficients because a DC coefficient in a bitstream is represented as the difference from the previous DC coefficient. On the other hand, a DC coefficient immediately after a restart marker is represented by its own value. Thus, inserting restart markers at a certain interval interrupts the error propagation among the DC coefficients in the encoded sequence. The restart interval (RI), which is an interval at which the restart marker is inserted, is set during JPEG encoding.

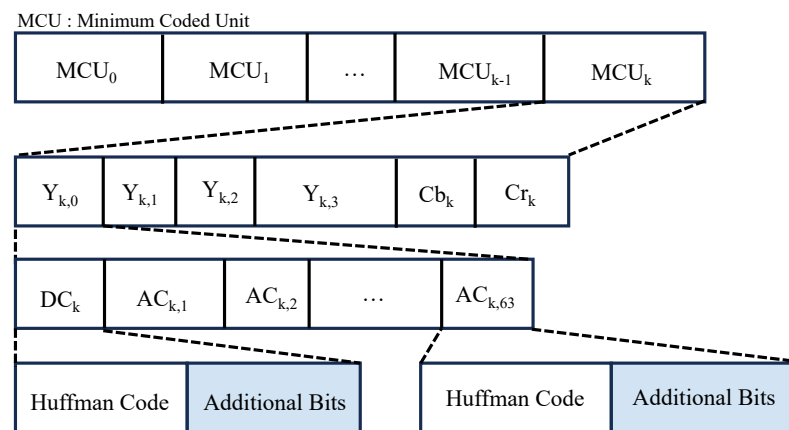


Figure 3. Structure of image data in JPEG bitstreams.

2.3. Bitstream-Level-Encryption Considering Marker Code Generation

Kobayashi et al. pointed out the requirements for being compliant with the JPEG format and keeping file sizes unchanged for bitstream-level JPEG encryption [16,17]. However, all bitstream-level JPEG encryption methods, including their method, cannot generate encrypted images with various modes of visibility such as partially encrypted images. Accordingly, we propose a novel bitstream-level encryption method for JPEG images that considers the use of the above requirements. An overview of the previous method [16,17] is given in Figure 4. The encryption procedure and requirements are summarized below.

- Step1: Generate a pseudo-random binary number (PRN) sequence consisting of 0 and 1 by using a secret key, K1.
- Step2: Analyze an input bitstream and extract bytes that satisfy the encryption requirements from the bitstream.
- Step3: Carry out exclusive-or (XOR) operations between the additional bits of the extracted bytes and the PRN sequence.
- Step4: Replace the additional bits with the XOR results.

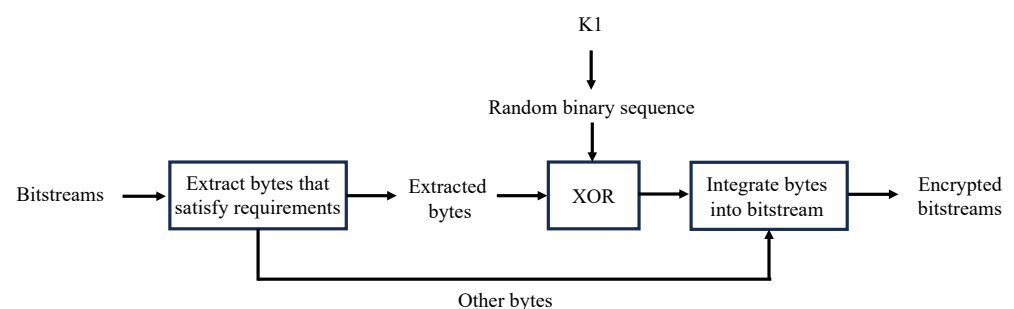


Figure 4. Encryption procedure of previous method.

The Huffman code shown in Figure 3 represents the range of each DCT coefficient value and the length of the additional bits. In contrast, the additional bits are used to identify a coefficient value from the range. If Huffman codes are encrypted, the bitstream is not compatible with the JPEG standard in general, so only additional bits are encrypted. When a new $FF_{(16)}$ is generated through the encryption process in image data, byte stuffing is conducted, and the file size increases. In contrast, when an existing $FF_{(16)}$ is lost through the encryption process, the JPEG decoder does not skip $FF_{(16)}$ following the initial $00_{(16)}$. Thus, to avoid changing the file size of JPEG images, bytes that satisfy the above requirements are extracted from a bitstream in Step 2. The requirements are summarized below.

When image data are divided into 1-byte units, the pattern of each byte is classified into five types as shown in Figure 5, where Case 4 satisfies the requirements for the file size of JPEG images. Each case is explained below.

- Case 1: It consists of only Huffman codes.
 Case 2: It consists of only additional bits.
 Case 3: It consists of Huffman codes and additional bits, and every bit in the Huffman code is 1.
 Case 4: It consists of Huffman codes and additional bits, and the Huffman code includes 0.
 Case 5: It consists of 0 only, and the byte is located following $FF_{(16)}$.

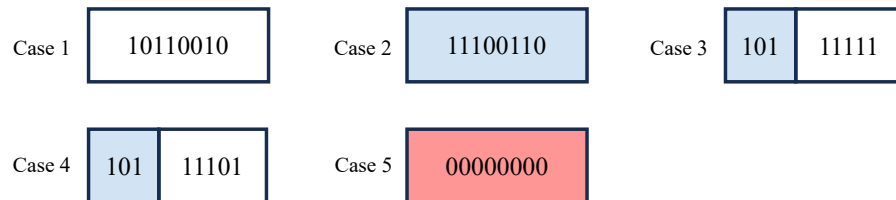


Figure 5. Five patterns of divided bytes. White, blue, and red regions represent Huffman codes, additional bits, and byte inserted by byte stuffing, respectively.

In Cases 1 and 5, the bytes do not contain any additional bits, so we do not encrypt them. In Case 2, the byte consists of additional bits only, so if the byte is encrypted, a new $FF_{(16)}$ might be generated unintentionally. Further, in Case 3, if all the additional bits are turned to 1 by encryption, $FF_{(16)}$ is generated.

Conversely, if all of the additional bits are 1 and the byte indicates $FF_{(16)}$, the encryption could cause 0 in the additional bits, resulting in the loss of the $FF_{(16)}$. Thus, the bytes should not be encrypted in those two cases. Consequently, we can encrypt additional bits in Case 4 to avoid not only generating $FF_{(16)}$ but also losing $FF_{(16)}$ due to encryption.

It can be concluded that the requirements used in Step 2 mean only selecting 1-byte units that apply to Case 4. The use of the requirements allows us not only to generate encrypted JPEG images with the same file size as that of JPEG images without encryption but to also use a standard JPEG decoder.

3. Proposed Method

The proposed method is explained here. The method allows us to generate encrypted images with various types of visibility including partially encrypted images by using restart markers while maintaining the same file size as that of JPEG images without encryption. The details are given below.

3.1. Encryption Using Extended-Block Permutation

As shown in Figure 6, we use restart (RST) markers for the first time, which can be inserted at regular intervals between MCUs for encryption so that a series of MCU blocks separated by RST markers is defined as an extended block. Figure 7a illustrates the encryption procedure of the proposed method, where a JPEG bitstream with RST markers is input.

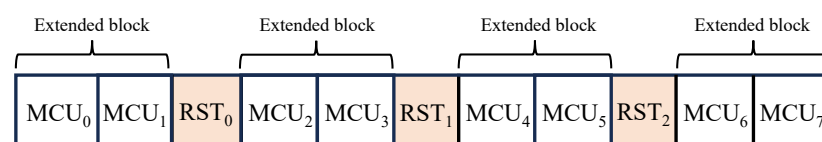


Figure 6. Bitstreams after insertion of restart markers.

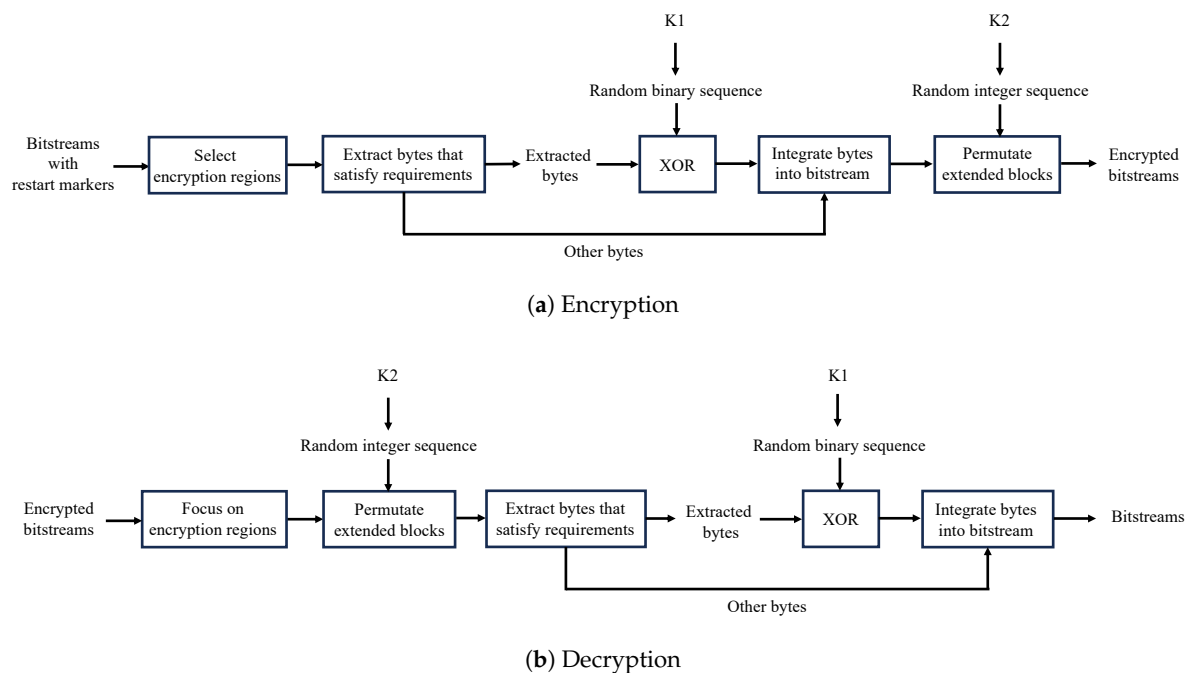


Figure 7. Procedure of proposed method.

- Step1: Select extended blocks to be encrypted and those to be encrypted and position-shuffled from an original image.
- Step2: Extract bytes that satisfy the encryption conditions described in Section 2.3 from the extended blocks selected in Step 1.
- Step3: Generate a PRN binary sequence by using secret key K1. Carry out an exclusive-or (XOR) operation between the PRN sequence and additional bits of the extracted bytes.
- Step4: Replace the additional bits with the XOR results.
- Step5: Permute the positions of the extended blocks selected in Step 1 by using secret key K2.

Figure 8 shows several practical examples of encrypted images after all the above steps have been applied. Note that Figure 8a,c were selected from the DeepLesion dataset [30] and Kodak Lossless True Color Image Suite [31], respectively. The differences between the method and the previous one [16,17] are in Step 1, Step 5, and the use of RST markers. A user first determines the regions of an image to be encrypted and then encrypts extended blocks that cover the determined regions. An extended block is composed of a series of MCUs separated by restart markers and is the smallest unit of encryption. The user can select arbitrary regions in units of extended blocks. Note that the encrypted regions cannot be selected in pixel units. The requirements in Step 2 are the same as those in [16,17].

3.1.1. Selection of Encrypted Regions

The proposed method can generate partially encrypted images that have both encrypted and unencrypted regions in them. In addition, the mode of visibility of the encrypted images can be selected as illustrated in Figure 9. This property of our method is made possible by using restart markers.

Figure 10 shows an example of a JPEG bitstream including RST markers, where the restart interval (RI) is the number of MCU blocks between RST markers. The method allows us to freely choose whether to encrypt each extended block or not. In the figure, blue and white regions represent encrypted and unencrypted blocks, respectively, and red regions are RST markers. By using RST markers, the effects of DPCM used for compressing DC coefficients can be prevented within each extended block. Our method uses RST

markers for bitstream-level JPEG encryption for the first time so that the effects of DPCM are prevented.

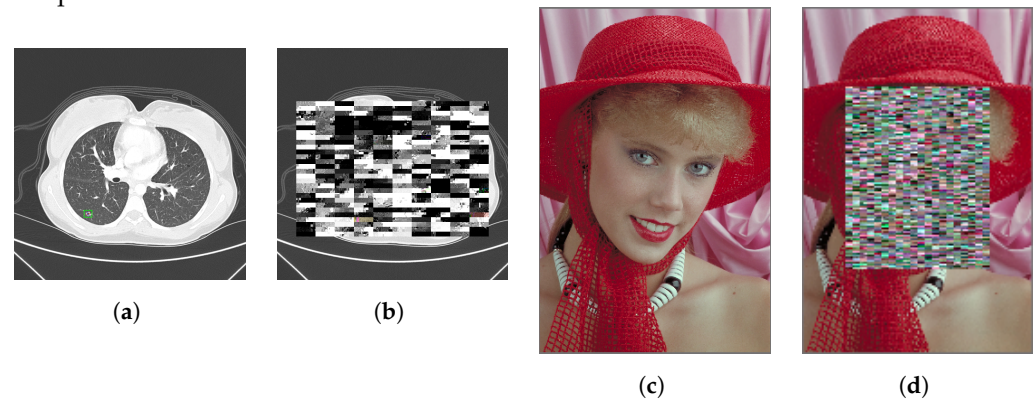


Figure 8. Example of encrypted images. (a) Medical image (768×768 pixels). (b) Encrypted medical image. (c) Image with face (2048×3072 pixels). (d) Encrypted image with face.

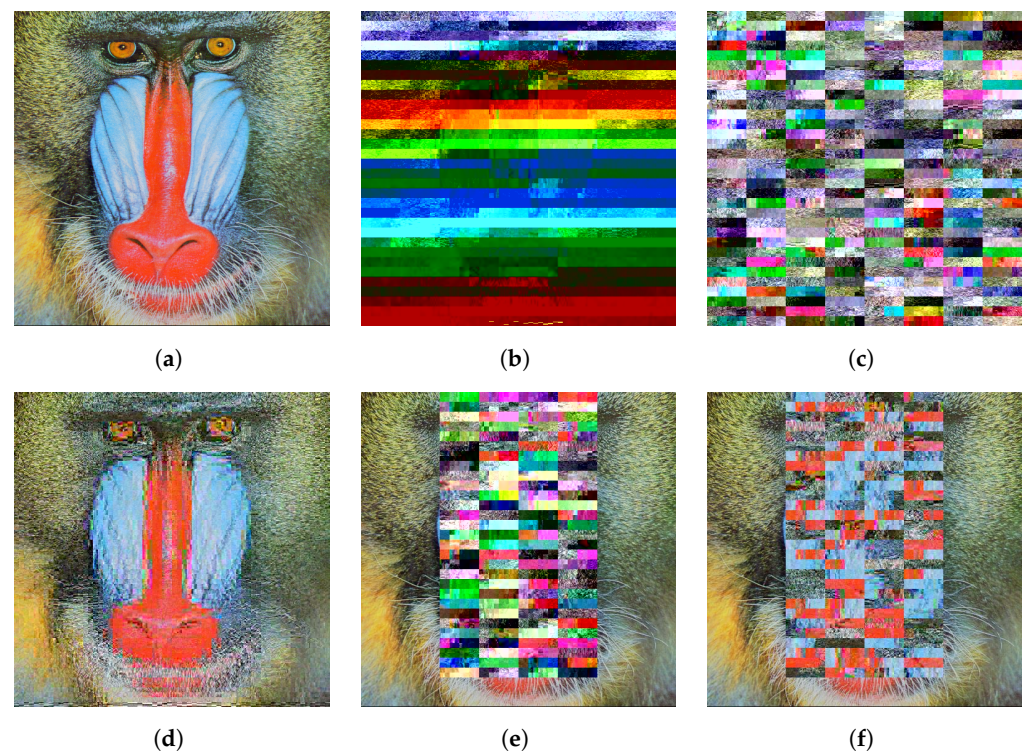


Figure 9. Example of encrypted images. (a) Original image. (b) Previous method. (c) Proposed method with block permutation. (d) Proposed method with only AC coefficients. (e) Proposed method with partial encryption and block permutation. (f) Proposed method with partial encryption for only AC coefficients and block permutation.

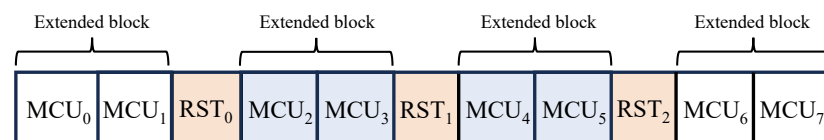


Figure 10. Example of encrypted and unencrypted regions. Red and blue regions represent RST markers and encrypted regions, respectively ($RI = 2$).

3.1.2. Position Scrambling of Extended Blocks

The proposed method can randomly permute the positions of extended blocks consisting of multiple MCUs. In contrast, the conventional method in [16,17] cannot permute them,

so the visibility of encrypted images is not low enough (see Figure 9b), so images encrypted with the conventional method still have some visual information from the original images. Accordingly, our method can provide images with enhanced privacy-preserving. Figure 11 depicts an example of the position scrambling of extended blocks, where the positions of all extended blocks are permuted in Figure 11a. In contrast, some of the extended blocks are permuted to generate partially encrypted images in Figure 11b. As a precondition, MCU positions within the extended block are not moved. AC and DC coefficients in each extended block are encoded independently. Since the restart interval is constant, all extended blocks contain the same number of MCUs. The first DC coefficient value that appears in each extended block is not the difference from the previous DC coefficient but the original value itself. Therefore, overflows never occur, even if the extended blocks are reordered randomly. Using Figure 11b as an example, the value of the first DC coefficient of MCU2 is not the difference from the last DC coefficient of MCU5 but the original coefficient value itself. As described above, a standard JPEG codec can decode the image.

Accordingly, the proposed method can provide images with various visibility modes as demonstrated in Sections 4.2 and 4.3.

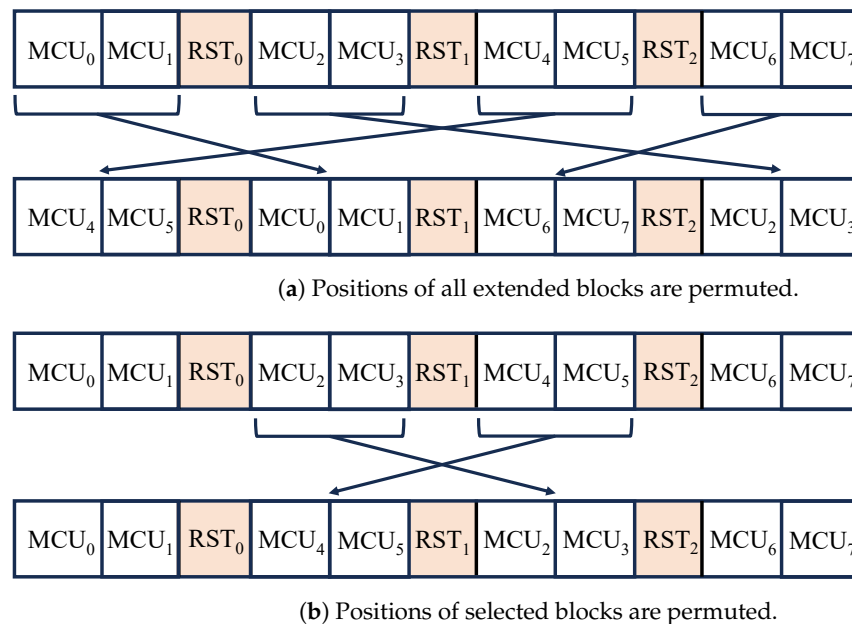


Figure 11. Example of bitstreams with permuted blocks.

3.1.3. Selection of Visibility Modes

The proposed method enables the visibility of encrypted images to be chosen from various modes as follows.

- We can divide an image into encrypted and unencrypted regions.
- We can freely choose whether to permute extended blocks in each encrypted region.
- We can freely choose whether to encrypt AC and DC coefficients in each extended block, respectively.
- We can choose a restart interval (*RI*) value.

A combination of the four selection items can be used when encrypting images, so we can choose the strength of visual protection for each region. Accordingly, our method allows us to freely choose convenience and security for each region.

By combining these options that a user can select, the user can control the intensity of partial encryption in terms of visual confidentiality and convenience. Applying more options from (a) to (c) leads to the enhancement of attack resistance. In particular, block scrambling in (b) is expected to significantly improve the attack resistance. However, the more options are applied, the higher the processing cost. In (d), the minimum value of *RI* is 1. The number cannot be greater than the total number of MCUs. Setting a small

value improves the attack resistance but increases the original file size. If fewer options are applied, the attack resistance is lower. However, even in such cases, it is expected to be effective in preventing unauthorized secondary use of the encrypted image.

3.2. Decryption Process

A JPEG bitstream encrypted with the proposed method can be decoded by using a standard JPEG decoder, but the visual information of the image is protected if the keys are not shared. The standard decoder does not have parameters for decryption such as the keys and encryption positions in the proposed method. Thus, the standard decoder cannot carry out any of the steps related to decryption. Only a user having these parameters can decrypt and decode the original image. To decode the original image, the user should decrypt the encrypted bitstream before decoding the image. Conversely, a user without these parameters can decode only the encrypted image by using the standard decoder. The procedure of decrypting an encrypted bitstream with secret keys is explained below (see Figure 7b).

- Step1: Select encrypted extended blocks from the encrypted image.
- Step2: Restore the position of the permuted extended blocks by using secret key K2.
- Step3: Extract bytes that satisfy the encryption conditions described in Section 2.3 from the extended blocks to be decrypted.
- Step4: Generate a PRN binary sequence by using secret key K1. Carry out an exclusive-or (XOR) operation between the PRN sequence and additional bits of the extracted bytes.
- Step5: Replace the additional bits with the XOR results.

Images decrypted with the above procedure are the same as those decoded from JPEG images without encryption.

3.3. Threat Model

The objective of an attacker is to recover visual information from encrypted JPEG images. We assume that the attacker has access to encrypted images and the encryption algorithm but does not possess the secret key. That is to say, we assume that the attacker can only carry out a cipher-text-only attack (COA) using encrypted images.

Several COAs have been proposed to restore visual information from encrypted images [6,7,11,16,32]. In addition, security analysis methods are used to evaluate the robustness of encryption methods against attacks in general. In this paper, we use a brute force attack, the sketch attack [6,7,11], and key sensitivity analysis [7,11,16] to evaluate the proposed method.

A key space is the total number of patterns that can be generated by a given encryption algorithm. There is an attack called the sketch attack, which attempts to obtain a rough sketch of an original image from its encrypted image. Here, we assume the non-zero-counting attack (NZCA), which is one type of sketch attack. Resistant image-encryption methods are required to be sensitive to even the slightest key change. If the key is changed by one bit, the decrypted image should be completely different from the original image.

The details on the security analysis will be given in Section 4.5, and the proposed method will be compared with other encryption methods for JPEG bitstreams. Our method has a similar performance to the conventional one [16,17], but it will be demonstrated to be more robust than the conventional one due to the use of restart markers.

4. Experimental Results and Discussion

In experiments, the effectiveness of the proposed method was evaluated in terms of the generation of partially encrypted images and the selection of visibility modes. In addition, the method was demonstrated to be a file-size-constant method.

4.1. Experimental Setup

In experiments, we used 24 test images with 2048×3072 pixels or 2048×3072 pixels from Kodak Lossless True Color Image Suite [31]. We used libjpeg [33] for JPEG compression, where the color subsampling and quality factor Q were set to 4:2:0 and 80, respectively. The restart interval RI was four or eight. PRN binary sequences were generated by using the pseudo-random number generator of HMAC_DRBG [34] with a 384-bit key.

4.2. Effects of Block Permutation

We first encrypted all areas of each image and permuted the position of all extended blocks to confirm the effects of block permutation. Figure 12 shows images encrypted with/without block permutation. The previous method [16,17] cannot carry out block permutation, but the proposed one can due to the use of RST markers. From the figure, the images encrypted without block permutation still had the outline of the original ones and information on colors. In contrast, the visibility of images encrypted with block permutation was much lower than that of the images without block permutation. From the result, the permutation of extended blocks was demonstrated to be effective in protecting the visibility of images.

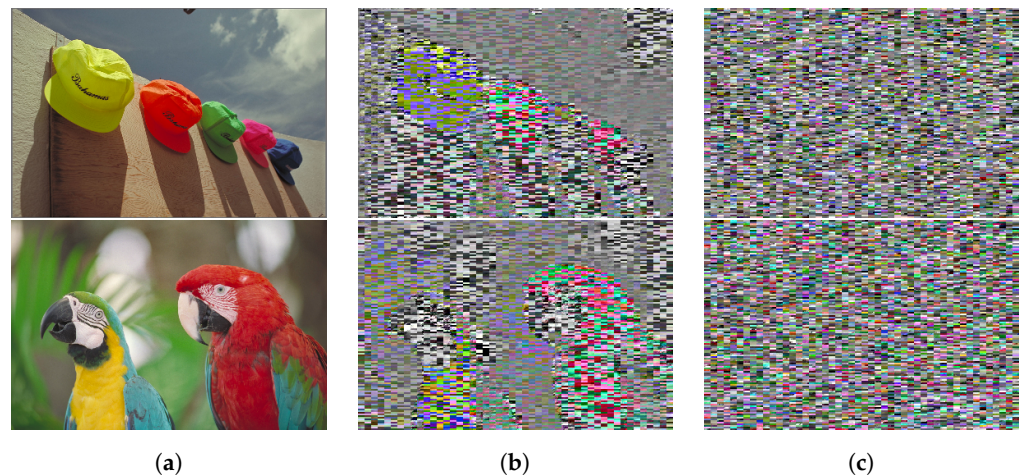


Figure 12. Effects of block permutation (kodim03 and kodim23). (a) Original image; (b) Encrypted image without block permutation ($RI = 4$); (c) Encrypted image with block permutation ($RI = 4$).

4.3. Generating Partially Encrypted Images

Next, partially encrypted images were generated by using our method. We defined encrypted regions as shown in Figure 13a. Figure 13 also shows an example of encrypted images without block permutation, but DC and AC coefficients in the encrypted region were encrypted. As shown in Figure 13b,c, partial-encrypted images were demonstrated to be generated by using our method without block permutation, so the method allows us to protect the visual information of sensitive regions. No conventional methods for JPEG bitstream-level encryption can generate such partially encrypted images.

Our method can generate encrypted images that have various types of visual information. One of the types is selected encrypted regions such that all regions were selected for encryption as in Figure 12, and a part of an image was selected in Figure 13. In addition, depending on whether or not block permutation is carried out, the visibility of encrypted images is changed as illustrated in Figures 13 and 14.

Our method provides more ways for changing the visibility of encrypted images. Figure 14 illustrates that an RI value is also a variable that can affect the visibility of encrypted images. In addition, Figure 15 shows other examples of encrypted images, where only AC coefficients are encrypted in Figure 15b, and the encryption of AC coefficients and block permutation are carried out in Figure 15c. From Figure 15b, when only AC coefficients are encrypted, the visual information of the images is not completely confidential. However,

the image quality within the red frame is degraded compared with the original image. Therefore, as shown in Figure 15b, the detailed features of the image such as small letters are difficult to see.

In contrast, when block permutation is carried out for encryption in addition to the encryption of AC coefficients, the visibility in Figure 15c is significantly degraded compared with Figure 15b. Since DC coefficients are not encrypted, Figure 15c shows no color tone change. Accordingly, the proposed method is verified not only to provide partially encrypted images but to also generate encrypted images with various types of visual information. Contrarily, the previous methods [4–9,16,17] have not been designed to partially encrypt an image, so it is difficult for them to accomplish partial encryption.

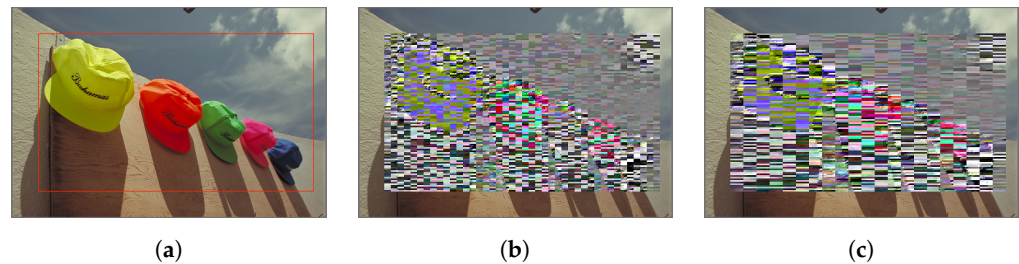


Figure 13. Partially encrypted images without block permutation (kodim03). (a) Target area for encryption; (b) Partially encrypted image ($RI = 4$); (c) Partially encrypted image ($RI = 8$).

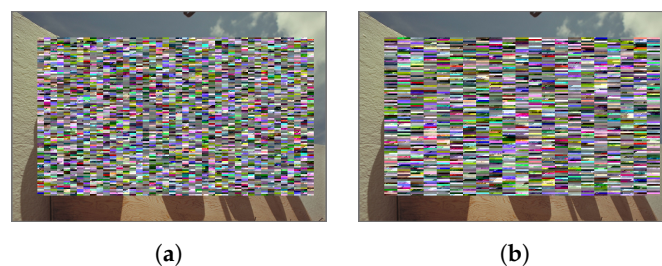


Figure 14. Partially encrypted images with block permutation. (a) Partially encrypted image ($RI = 4$); (b) Partially encrypted image ($RI = 8$).

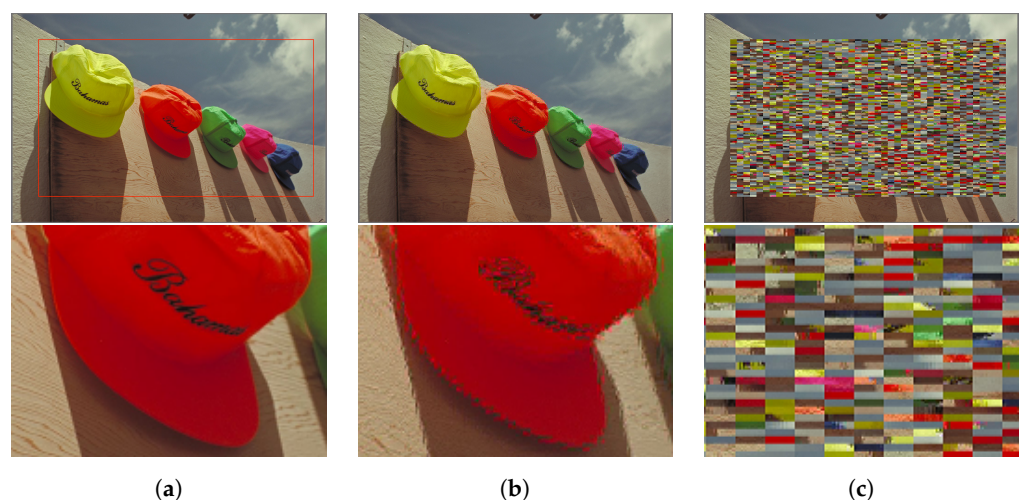


Figure 15. Partially encrypted images with encryption of only AC coefficients (kodim03, $RI = 4$). Zoom-in of the boxed region is shown in bottom of each image. (a) Target area for encryption; (b) Partially encrypted image without block permutation; (c) Partially encrypted image with block permutation.

4.4. File Size Preserving

Finally, we demonstrate that the proposed method can generate encrypted JPEG images with the same file size as JPEG images without encryption. The eight images in Figure 16 are examples of images used in the experiment, and 24 images were used for evaluation in total. Experiment results are given in Table 1, where the proposed method uses block permutation, and the encryption of DC and AC coefficients is carried out under $RI = 4$.

From the table, the file size of the JPEG images is confirmed to be unchanged before and after encryption. We also confirm that the file size of all 24 encrypted images is the same as that of the JPEG images without encryption. Consequently, our encryption process does not affect the file size of JPEG images.

Furthermore, Table 2 shows the change in file size of three images, Lena, Mandrill, and Peppers, for each method, where the images are encoded with $Q = 80$. From the table, for the previous bitstream-level methods (Type 4) [16,17], the size of all the images is equal to that of the original files without encryption. In contrast, for Cheng et al.'s method [5] (Type 4), the file size was slightly changed due to the generation and loss of $FF_{(16)}$ caused by block replacement. In the case of the occurrence of $FF_{(16)}$, $00_{(16)}$ was inserted, resulting in an increase in the file size. On the other hand, in the case of the loss of $FF_{(16)}$, the file size decreased because the $00_{(16)}$ immediately after $FF_{(16)}$ was deleted to correctly handle the encoded sequence. Similarly, the methods of [4,6–9] never ensure that the encrypted file size will be exactly matched to the original file size. For encryption-then-compression (EtC) (Type 2) [1], the file size was also changed due to the use of a decoder and encryption, where only block scrambling was performed in 16×16 units for encryption. By reordering blocks, the difference value of the DC coefficients becomes larger, resulting in a change in file size. Accordingly, when only bitstream-level encryption was applied to byte units carefully selected in consideration of marker code generation, the file size of the JPEG images was not changed. In addition, the proposed method allows us to generate encrypted images with various visibility modes including partially encrypted images, which are important for convenience and security, while maintaining the file size-preserving property.

Table 1. File size before and after encryption.

Image	Before Encryption [Bytes]	After Encryption [Bytes]
kodim03	570,731	570,731
kodim04	738,287	738,287
kodim10	843,457	843,457
kodim12	529,742	529,742
kodim14	841,642	841,642
kodim18	1,259,498	1,259,498
kodim23	592,913	592,913
kodim24	749,104	749,104

Table 2. File size before and after encryption for each method ([bytes]).

Image	Lena	Mandrill	Peppers
Original	44,293	89,057	48,671
Previous method [16,17]	44,293	89,057	48,671
Cheng [5]	44,275	89,059	48,670
EtC [1]	45,300	90,109	49,739



Figure 16. Example of test images.

4.5. Security Analysis

The security robustness of the proposed method depends on the options defined in Section 3.1.3. Here, we analyzed the proposed method in the case that all DCT coefficients in the entire image were encrypted for the entire image, and the positions of all extended blocks were permuted. As an example, we show the result of the security analysis in the case of Figure 16g.

The encryption algorithm proposed in this paper consists of the encryption of additional bits in bytes that satisfy the conditions and position permutation of extended blocks. We suppose that JPEG encoding is applied to an $M \times N$ -pixel image with $RI = r$ and 4:2:0 color subsampling. First, in the case where the bytes to be encrypted are T bytes, the minimum and maximum sizes of the key space resulting from encrypting the additional bits S_{enc_min} and S_{enc_max} are expressed by

$$S_{enc_min} = 2^T, \quad (1)$$

$$S_{enc_max} = 2^{7T}, \quad (2)$$

since the number of bits to be encrypted in each of T bytes is at least one bit and at most seven bits. The key space resulting from the permutation of the extended block S_{bp} is given by

$$S_{bp} = \left[\left(\left\lceil \frac{M}{16} \right\rceil \times \left\lceil \frac{N}{16} \right\rceil \times \frac{1}{r} \right) \right]!. \quad (3)$$

Consequently, the overall minimum key space S_{min} and maximum key space S_{max} are obtained by

$$S_{min} = S_{enc_min} \times S_{bp} = 2^T \times \left[\left(\left\lceil \frac{M}{16} \right\rceil \times \left\lceil \frac{N}{16} \right\rceil \times \frac{1}{r} \right) \right]!, \quad (4)$$

$$S_{max} = S_{enc_max} \times S_{bp} = 2^{7T} \times \left[\left(\left\lceil \frac{M}{16} \right\rceil \times \left\lceil \frac{N}{16} \right\rceil \times \frac{1}{r} \right) \right]!. \quad (5)$$

Assuming a 512×512 -pixel image, $RI = 4$, and 55,206 bytes to be encrypted, the key space is $2^{55,206} \times 256!$. Thus, the proposed method provides a large key space. As the restart interval becomes shorter, the key space becomes larger since the number of extended blocks increases. In the case of partial encryption, the numbers of bytes to be encrypted and extended blocks to be permuted are limited, so the key space is smaller.

NZCA was performed on the luminance component of the encrypted image. As shown in Figure 17, in the conventional method [16,17], the original image is revealed in its outline. In comparison, the proposed method completely conceals the original image. This is achieved by replacing enlarged blocks. The proposed method is as robust as other methods that use coefficient replacement [6,7,11] since it includes the replacement of extended blocks.

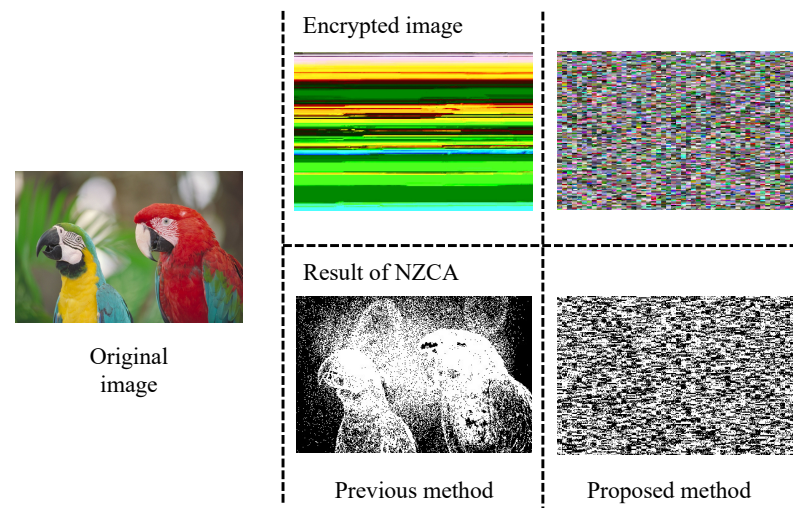


Figure 17. Results of non-zero-counting attack (NZCA).

We decrypted the image in Figure 12c with incorrect keys that were changed by one bit from the original keys K1 and K2. The result is shown in Figure 18. This confirms that the decryption was not successful at all and that the image content remained confidential. The peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM) values for Figure 18b,c relative to Figure 18a were 9.28 dB/0.3410 and 9.25 dB/0.3414, respectively. It is clear that when using a key that is incorrect by even one bit, the encrypted image cannot be decrypted at all. We also confirmed that the original image in Figure 18a was perfectly retrieved when using correct keys.

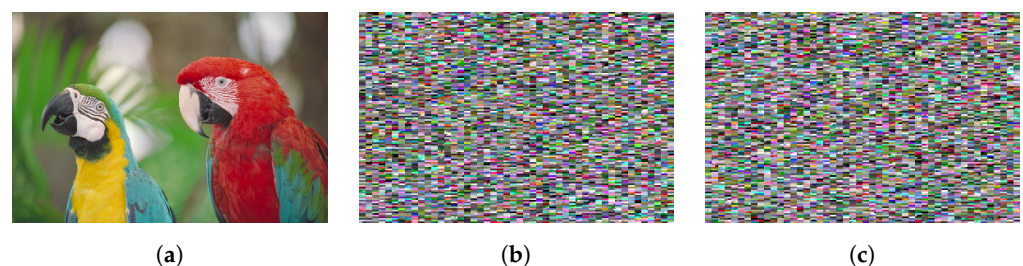


Figure 18. Key sensitivity. (a) Original image; (b) Encrypted image ($RI = 4$, PSNR = 9.28 dB, SSIM = 0.3410); (c) Decrypted image with incorrect keys ($RI = 4$, PSNR = 9.25 dB, SSIM = 0.3414).

In this experiment, we assumed that an attacker carried out COAs. We evaluated the attack resistance of the proposed method using typical attack methods in terms of the difficulty in recovering visual information. The analysis results showed that the proposed method was as resistant to the attacks as other methods. In addition, the proposed method is more attack resistant than the conventional method [16,17] owing to the permutation of extended blocks. Although an example was shown here, we confirmed that the analogous trends were obtained for the other images.

5. Conclusions

We proposed a novel bitstream-level encryption method for JPEG images. The method allows us to generate partially encrypted images that conventional methods for JPEG bitstream-level encryption cannot. In addition, it can generate encrypted images with various types of visual information while maintaining the same file size as that of JPEG images without encryption. To achieve this, we proposed using a code called RST marker for the first time. In experiments, the method was compared with the state of the art, and the effectiveness of the method was verified in terms of file size preserving and the visibility of encrypted images. We plan to extend the method to other file formats such as video coding in future work.

Author Contributions: Conceptualization, M.H., S.I. and H.K.; methodology, M.H. and H.K.; validation, M.H. and S.I.; investigation, M.H.; writing—original draft preparation, M.H.; writing—review and editing, S.I. and H.K.; supervision, S.I. and H.K.; project administration, S.I. and H.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by JSPS KAKENHI, grant number JP21H01327.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Kurihara, K.; Kikuchi, M.; Imaizumi, S.; Shiota, S.; Kiya, H. An encryption-then-compression system for JPEG/Motion JPEG standard. *IEICE Trans. Fundam.* **2015**, *E98-A*, 2238–2245. [\[CrossRef\]](#)
2. Ahmad, I.; Shin, S. IIB-CPE: Inter and Intra Block Processing-Based Compressible Perceptual Encryption Method for Privacy-Preserving Deep Learning. *Sensors* **2022**, *22*, 8074. [\[CrossRef\]](#) [\[PubMed\]](#)
3. Ahmad, I.; Shin, S. A Perceptual Encryption-Based Image Communication System for Deep Learning-Based Tuberculosis Diagnosis Using Healthcare Cloud Services. *Electronics* **2022**, *11*, 2514. [\[CrossRef\]](#)
4. Niu, X.; Zhou, C.; Ding, J.; Yang, B. JPEG encryption with file size preservation. In Proceedings of the International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), Harbin, China, 15–17 August 2008; pp. 308–311.
5. Cheng, H.; Zhang, X.; Yu, J.; Zhang, Y. Encrypted JPEG image retrieval using block-wise feature comparison. *J. Vis. Commun. Image Represent.* **2016**, *40*, 111–117. [\[CrossRef\]](#)
6. He, J.; Huang, S.; Tang, S.; Huang, J. JPEG Image Encryption with Improved Format Compatibility and File Size Preservation. *IEEE Trans. Multimed.* **2018**, *20*, 2645–2658. [\[CrossRef\]](#)
7. Qin, C.; Hu, J.; Li, F.; Qian, Z.; Zhang, X. JPEG Image Encryption with Adaptive DC Coefficient Prediction and RS Pair Permutation. *IEEE Trans. Multimed.* **2023**, *25*, 2528–2542. [\[CrossRef\]](#)
8. Yuan, Y.; He, H.; Yang, Y.; Mao, N.; Chen, F.; Ali, M. JPEG image encryption with grouping coefficients based on entropy coding. *J. Vis. Commun. Image Represent.* **2023**, *97*, 103975. [\[CrossRef\]](#)
9. Unterwieser, A.; Uhl, A. Length-preserving Bit-stream-based JPEG Encryption. In Proceedings of the on Multimedia and security (MM&Sec), New York, NY, USA, 6–7 September 2012; pp. 85–90.
10. Khan, M.I.; Jeoti, V.; Khan, M.A. Perceptual encryption of JPEG compressed images using DCT coefficients and splitting of DC coefficients into bitplanes. In Proceedings of the International Conference on Intelligent and Advanced Systems (ICIAS), Kuala Lumpur, Malaysia, 15–17 June 2010; pp. 1–6.
11. Peng, Y.; Fu, C.; Cao, G.; Song, W.; Chen, J.; Sham, C. W. JPEG-compatible Joint Image Compression and Encryption Algorithm with File Size Preservation. *ACM Trans. Multimed. Comput. Commun. Appl.* **2024**, *20*, 105. [\[CrossRef\]](#)
12. Shimizu, K.; Suzuki, T. Finely Tunable Bitcuboid-Based Encryption With Exception-Free Signed Binarization for JPEG Standard. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 4895–4908. [\[CrossRef\]](#)
13. Li, P.; Lo, K.T. Joint image encryption and compression schemes based on 16×16 DCT. *J. Vis. Commun. Image Represent.* **2019**, *58*, 12–24. [\[CrossRef\]](#)
14. Li, P.; Sun, Z.; Situ, Z.; He, M.; Song, T. Joint JPEG Compression and Encryption Scheme Based on Order-8-16 Block Transform. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 7687–7696. [\[CrossRef\]](#)
15. Cao, W.; Leng, X.; Yu, T.; Gu, X.; Liu, Q. A Joint Encryption and Compression Algorithm for Multiband Remote Sensing Image Transmission. *Sensors* **2023**, *23*, 7600. [\[CrossRef\]](#) [\[PubMed\]](#)
16. Kobayashi, H.; Kiya, H. Bitstream-based jpeg image encryption with file-size preserving. In Proceedings of the IEEE Global Conference on Consumer Electronics (GCCE), Nara, Japan, 9–12 October 2018; pp. 384–387.
17. Kobayashi, H.; Kiya, H. File-Size Preserving Encryption of JPEG Images in the Bitstream Domain. *IEICE Trans. Inf. Syst. Jpn. Ed.* **2019**, *J102-D*, 787–795.

18. Ra, M.-R.; Govindan, R.; Ortega, A. P3: Toward privacy-preserving photo sharing. In Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI), Lombard, IL, USA, 2–5 April 2013; pp. 515–528.
19. Lee, T.H.; Hsu, H.H.; Chang, P.C. Restart marker regulation technique for progressive JPEG image coding in mobile communications. *IEEE Commun. Lett.* **2000**, *4*, 411–413.
20. Hafsa, A.; Sghaier, A.; Malek, J.; Machhout, M. Image encryption method based on improved ECC and modified AES algorithm. *Multimed. Tools Appl.* **2021**, *80*, 19769–19801. [[CrossRef](#)]
21. Lin, C.-H.; Hu, G.-H.; Chan, C.-Y.; Yan, J.-J. Chaos-Based Synchronized Dynamic Keys and Their Application to Image Encryption with an Improved AES Algorithm. *Appl. Sci.* **2021**, *11*, 1329. [[CrossRef](#)]
22. Yu, P.; Tang, J.; Xia, Z.; Li, Z.; Weng, J. A Privacy-Preserving JPEG Image Retrieval Scheme Using the Local Markov Feature and Bag-of-Words Model in Cloud Computing. *IEEE Trans. Cloud Comput.* **2023**, *11*, 2885–2896. [[CrossRef](#)]
23. Zhou, J.; Liu, X.; Au, O.C.; Tang, Y.Y. Designing an efficient image encryption-then-compression system via prediction error clustering and random permutation. *IEEE Trans. Inf. Forensics Secur.* **2014**, *9*, 39–50. [[CrossRef](#)]
24. Puchala, D.; Stokfiszewski, K.; Yatsymirskyy, M. Encryption before compression coding scheme for JPEG image compression standard. In Proceedings of the Data Compression Conference (DCC), Snowbird, UT, USA, 24–27 March 2020; pp. 313–322.
25. Kiya, H.; Maung, A.P.M.; Kinoshita, Y.; Imaizumi, S.; Shiota, S. An overview of compressible and learnable image transformation with secret key and its applications. *APSIPA Trans. Signal Inf. Process.* **2022**, *11*, e11. [[CrossRef](#)]
26. Hamano, G.; Imaizumi, S.; Kiya, H. Effects of JPEG Compression on Vision Transformer Image Classification for Encryption-then-Compression Images. *Sensors* **2023**, *23*, 3400. [[CrossRef](#)] [[PubMed](#)]
27. Mitmproxy. Available online: <https://mitmproxy.org> (accessed on 7 March 2024).
28. Google Photos. Available online: <https://photos.google.com/> (accessed on 7 March 2024).
29. ISO/IEC IS-10918-1; International Organization for Standardization. Information technology—Digital Compression and Coding of Continuous-Tone still Images: Requirements and Guidelines. International Organization for Standardization: Geneva, Switzerland, 1994.
30. Yan, K.; Wang, X.; Lu, L.; Summers, R.M. DeepLesion: Automated Mining of Large-Scale Lesion Annotations and Universal Lesion Detection with Deep Learning. *J. Med. Imaging* **2018**, *5*, 036501. [[CrossRef](#)] [[PubMed](#)]
31. Kodak Lossless True Color Image Suite. Available online: https://www.math.purdue.edu/~lucier/PHOTO_CD/BMP_IMAGES/ (accessed on 7 March 2024).
32. Chang, A.H.; Case, B.M. Attacks on image encryption schemes for privacy-preserving deep neural networks. *arXiv* **2020**, arXiv:2004.13263.
33. Independent JPEG Group. Available online: <https://www.ijg.org> (accessed on 7 March 2024).
34. Barker, E.B.; Kelsey, J.M. SP 800-90A. Recommendation for Random Number Generation Using Deterministic Random Bit Generators; NIST: Gaithersburg, MD, USA, 2012.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.