A Block-Permutation-Based Image Encryption Allowing Hierarchical Decryption

Yusuke IZAWA*, Shoko IMAIZUMI* and Hitoshi KIYA†
* Chiba University, Chiba, Japan
E-mail: izawa@chiba-u.jp, imaizumi@chiba-u.jp
† Tokyo Metropolitan University, Tokyo, Japan
E-mail: kiya@tmu.ac.jp

***Abstract*—This paper proposes a block-permutation-based encryption (BPBE) scheme, which allows only decrypting particular regions in the encrypted image. It is difficult to perform partial decryption in the conventional scheme, because it encrypts the entire image at once. By composing regions in the original image, we can conduct the hierarchical encryption and achieve the partial decryption in the proposed scheme. Additionally, the proposed scheme can maintain the JPEG-LS compression efficiency of the encrypted images compared to the conventional scheme. Moreover, the resilience against jigsaw puzzle solving problems can be enhanced by applying the proposed scheme to the combined images. We further consider an efficient key management by using hash chains.**

I. Introduction

With the spread of smartphones and tablet terminals, the number of images in social networking services, cloud services, and so forth, has also increased. On the other hand, the privacy and copyright protection issues relevant to the images have been seriously concerned. Compression-then-Encryption (hereafter, CtE) systems, which implement compression before encryption, are the traditional frameworks to transmit images. However, the image content is exposed to a service provider in the CtE systems. For this reason, Encryption-then-Compression (hereafter, EtC) systems have been studied to make the transmission more secure [1]–[4]. An image owner carries out encryption for own images before transmission in the EtC systems. Block-Permutation-Based Encryption (hereafter, BPBE) schemes have been developed as the appropriate algorithms for EtC systems [5]–[9].

BPBE schemes can maintain the compression efficiency of their encrypted images relative to that of the original images. Those schemes first divide the original image into the same-sized blocks and execute four processes: positional scrambling, block rotation/flip, negative-positive transformation, and color component shuffling. Each process is, however, simultaneously conducted in the entire image using an encryption key. Therefore, it is difficult to retrieve a partial region of the encrypted image.

In the use of BPBE, we need to consider the resilience against Jigsaw Puzzle Solvers (hereafter, JPSs) [10]–[12] and brute-force attacks, where malicious users attempt to restore the original image from the encrypted image. JPS is an attack that aims to assemble the original image from a lot of pieces using the correlation among them. Since a BPBE image consists of multiple blocks, it could be possibly decrypted to the original image by JPS. It has been proved that the appropriate block size could improve the resilience against JPS attacks in BPBE [13]. However, it is also reported that the BPBE images, where the R, G, and B components are transformed identically, can be partially reconstructed by the conventional JPS in case that the number of the blocks is not enough. In order to improve the resilience against JPSs, the BPBE scheme, where the color components are transformed independently, have been proposed [8]. The encrypted images produced by this scheme [8] have wider color distributions than those produced by the previous schemes. Comparing the compression efficiency using JPEG-LS, the BPBE images, where the R, G, and B components are independently transformed, show almost the same compression efficiency as those, where the R, G, and B components are identically transformed.

In this paper, we propose an adaptable BPBE scheme, which allows only retrieving particular regions from the encrypted image without decrypting other regions. Our scheme first divides an image into several regions and subdivides each region into multiple blocks. Next, the three processes, that is, block rotation/flip, negative-positive transformation, and color component shuffling are performed region by region. Finally, we shuffle the positions of the blocks within the entire image by positional scrambling. Consequently, we can retrieve particular regions from the encrypted image. Our scheme can be applied a concatenated image, which consists of several images, to enhance the color distribution against JPSs. Moreover, we verify that the proposed scheme maintains the JPEG-LS compression efficiency relative to the conventional scheme. We further discuss a superior key management using hash chains.

II. Block-Permutation-based-Encryption [5], [8]

Figure 1 shows a BPBE procedure [5], [8]. BPBE divides an original image into fixed-size blocks and performs four processes: positional scrambling, block rotation and flip, negative-positive transformation, and color component shuffling. After the above processes, all the blocks are integrated into a single encrypted image. The encryption procedure is described as follows.

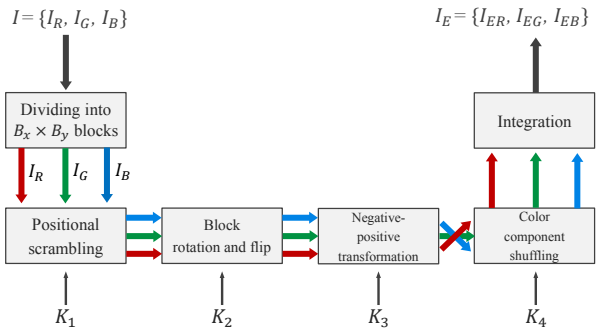


Fig. 1 Procedure of BPBE [5], [8].

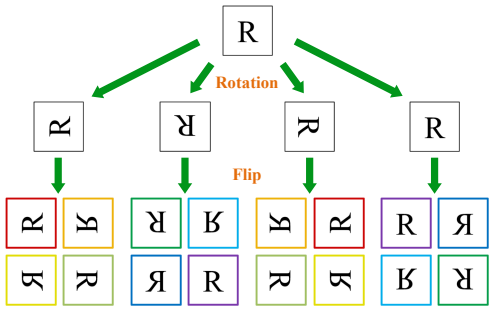


Fig. 2 Block rotation and flip.

Step1: Divide image $I = \{I_R, I_G, I_B\}$ with $M \times N$ pixels into multiple blocks with $B_x \times B_y$ pixels.
Step2: Randomly permute the position of each block using key K_1.
Step3: Rotate and flip each block using key K_2.
Step4: Apply the negative-positive transformation to each block using key K_3.
Step5: Shuffle the R, G, and B components in each block using key K_4.
Step6: Integrate all the blocks and generate encrypted image $I_E = \{I_{ER}, I_{EG}, I_{EB}\}$.

We clarify the main four processes in more detail below.

A. Positional Scrambling

The positions of the divided blocks are shuffled according to three random numbers generated from key $K_1 = \{K_{1,R}, K_{1,G}, K_{1,B}\}$. Note that the R, G, and B components in each block are shuffled by using K_1 identically [5], or independently [8].

B. Block Rotation and Flip

Each block is rotated 0, 90, 180, or 270 degrees and flipped horizontally and/or vertically or is not flipped using three random numbers generated from key $K_2 = \{K_{2,R}, K_{2,G}, K_{2,B}\}$. Figure 2 illustrates the procedure of rotation and flip. Each of rotation and flip has four operation types. There are 16 combinations when simply combining those two processes. Some combinations, however, correspond to other combinations. Accordingly, the number of total block patterns is eight. Note that the R, G, and B components in each block are rotated and flipped by using K_2 identically [5], or independently [8].

C. Negative-Positive Transformation

The negative-positive transformation reverses all of the pixel values in a block depending on three random numbers generated from key $K_3 = \{K_{3,R}, K_{3,G}, K_{3,B}\}$. In this operation, the transformed pixel value p' is given by

$$p' = \begin{cases} p & (r_{NPT}(i) = 0) \\ 255 - p & (r_{NPT}(i) = 1), \end{cases} \quad (1)$$

where p denotes the original pixel value and $r_{NPT}(i)$ is a random number for the i-th block, which is generated from key K_3. Note that the R, G, and B components in each block are transformed by using K_3 identically [5], or independently [8].

D. Color Component Shuffling

The color component shuffling permutes the R, G, and B components in each block in accordance with a random number generated from key K_4. In this operation, a set of the transformed color component c' is obtained by

$$c' = \begin{cases} \{c_R, c_G, c_B\} & (r_{CCS}(i) = 0) \\ \{c_R, c_B, c_G\} & (r_{CCS}(i) = 1) \\ \{c_G, c_R, c_B\} & (r_{CCS}(i) = 2) \\ \{c_G, c_B, c_R\} & (r_{CCS}(i) = 3) \\ \{c_B, c_R, c_G\} & (r_{CCS}(i) = 4) \\ \{c_B, c_G, c_R\} & (r_{CCS}(i) = 5), \end{cases} \quad (2)$$

where c_R, c_G, and c_B denote the R, G, and B components of the original image, respectively, and $r_{CCS}(i)$ is a random number for the i-th block, which is generated from key K_4.

The conventional scheme divides an original image into multiple blocks directly and encrypts the image based on them at once. Therefore, in the decryption process, all the blocks should be decrypted simultaneously. In other words, it is difficult to obtain the partially decrypted image according to the user rights. In the next section, we propose an adaptable BPBE, which flexibly allows partial decryption.

III. Proposed Methods

The proposed BPBE schemes have a hierarchical structure and enable partial decryption according to the user rights.

A. Proposed Encryption Procedure for Single Image

Figure 3 illustrates the procedure of the proposed scheme for a single image. This scheme first divides an image into several regions $R(i)$, where i is a region number, and subdivides each region into multiple blocks. Next, block rotation and

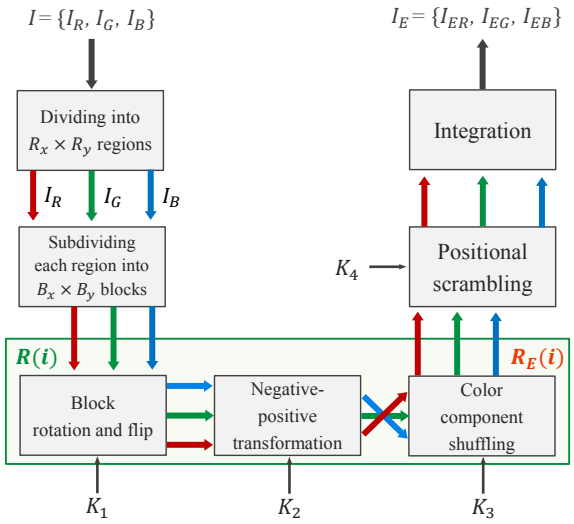


Fig. 3 Proposed encryption procedure for single image.

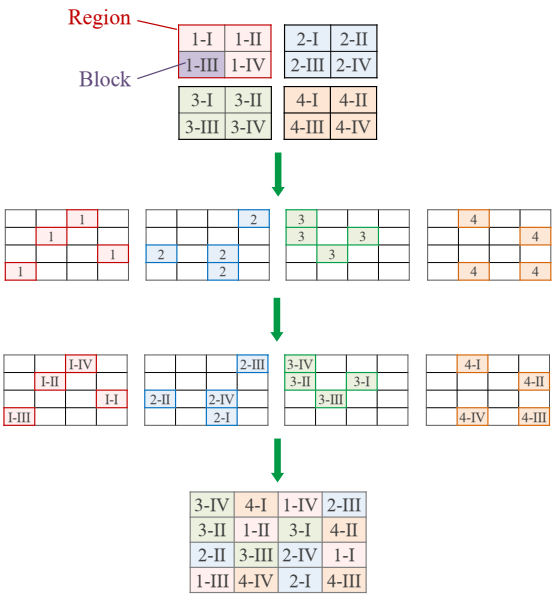


Fig. 4 Positional scrambling in proposed scheme.

flip, negative-positive transformation, and color component shuffling are applied to each region individually. After the above three scrambling processes, the positions of all the blocks are shuffled within the entire image. Eventually, the blocks are integrated into a single encrypted image.

Figure 4 shows the positional scrambling process in the proposed scheme. As shown in Fig. 4, the block positions of each region by this process are assigned by using key K_4 in advance. Accordingly, it is possible to retrieve the arbitrary regions from the encrypted image without decrypting the others. The steps in the proposed scheme are outlined below.

Step1: Divide original image I into multiple regions with $R_x \times R_y$ pixels. The i-th region is defined as $R(i) = \{R_{i,R}, R_{i,G}, R_{i,B}\}$, where i = 1, 2, ..., N is a region number.
Step2: Divide region $R(i)$ into $B_x \times B_y$ sized blocks.
Step3: Rotate and flip each block using key $K_{1,i}$.
Step4: Apply the negative-positive transformation to each block using key $K_{2,i}$.
Step5: Shuffle the R, G, and B components in each block using key $K_{3,i}$.
Step6: Integrate all the blocks and generate encrypted region $R_E(i) = \{R_{i,ER}, R_{i,EG}, R_{i,EB}\}$.
Repeat steps 2-6 for all the regions.

Step7: Assign the block positions of the positional scrambling to each region and shuffle the blocks depending on the assigned positions using key $K_{4,i}$ (Fig. 4).
Step8: Integrate all the blocks and generate encrypted image $I_E = \{I_{ER}, I_{EG}, I_{EB}\}$.

Note that the R, G, and B components in each block are transformed identically or individually in steps 3, 4, and 7. When the color components in each block are identically transformed, it can exactly maintain the JPEG compression efficiency of the encrypted image compared to that of the original image. However, the color distribution of the original image strongly affects that of its encrypted image. If the color components in each block are independently transformed, the encrypted image exhibits a much wider color distribution than the original image but its JPEG compression efficiency would be slightly degraded compared to the original image.

B. Proposed Encryption Procedure for Combined Image

In order to make the color distribution of the encrypted image wider, we propose to combine multiple images before encryption. By compounding blocks from multiple images, which have different color distributions, the encrypted image can possess a wider color distribution. We can also retrieve arbitrary images without decrypting other ones by applying the proposed scheme. Figure 5 illustrates the procedure of this scheme. It is assumed that we concatenate four different images in this paper. The procedure is explained as follows.

Step1: Divide four individual images with $M \times N$ pixels into $B_x \times B_y$ sized blocks individually. The i-th image is defined as $I(i) = \{I_{i,R}, I_{i,G}, I_{i,B}\}$, where i = 1, 2, 3, 4.
Step2: Rotate and flip each block using key $K_{1,i}$.

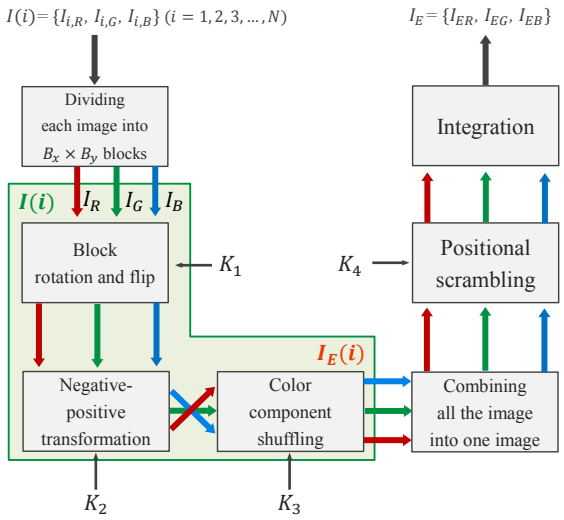


Fig. 5 Proposed encryption procedure for combined image.

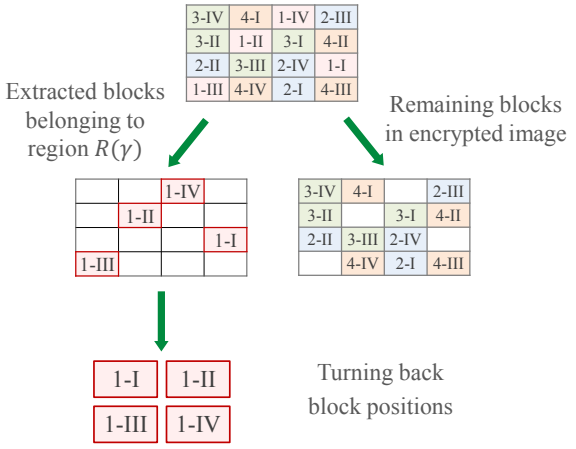


Fig. 6 Turning back block positions (γ = 1).

Step3: Apply the negative-positive transformation to each block using key $K_{2,i}$.
Step4: Shuffle the R, G, and B components in each block using key $K_{3,i}$.
Step5: Integrate all the blocks and generate encrypted image $I_E(i) = \{I_{i,ER}, I_{i,EG}, I_{i,EB}\}$.
Repeat steps 2-5 for all the images.

Step6: Assign the block positions of the positional scrambling to each image and shuffle the blocks depending on the assigned positions using key $K_{4,i}$.
Step7: Integrate all the blocks and generate encrypted image $I_E = \{I_{ER}, I_{EG}, I_{EB}\}$ with $2M \times 2N$ pixels.

C. Hierarchical Decryption Procedure

We elaborate the hierarchical decryption procedure, which only retrieves particular regions from the encrypted image. This algorithm is applicable to both schemes for a single image and a combined image. Here, we take a single image as an example to only decrypt the γ-th region.

Step1: Divide encrypted image I_E with $M \times N$ pixels into $B_x \times B_y$ sized blocks.
Step2: Turn back the block positions, which belong to region $R(\gamma)$ using key $K_{4,\gamma}$ as shown in Fig. 6.
Step3: Re-shuffle the R, G, and B components in each block using key $K_{3,\gamma}$.
Step4: Apply the negative-positive transformation to each block using key $K_{2,\gamma}$.
Step5: Rotate and flip each block using key $K_{1,\gamma}$
Step6: Integrate all the blocks and generate decrypted region $R_D(\gamma) = \{R_{\gamma,DR}, R_{\gamma,DG}, R_{\gamma,DB}\}$.

Note that it is possible to retrieve multiple regions by repeating steps 2-6.

D. Key Derivation Using Hash Chains

In case that the R, G, and B components in each block are transformed independently, the conventional scheme [8] requires twelve encryption keys, that is, $K_{1,R}$, $K_{1,G}$, $K_{1,B}$, $K_{2,R}$, $K_{2,G}$, $K_{2,B}$, $K_{3,R}$, $K_{3,G}$, $K_{3,B}$, $K_{4,R}$, $K_{4,G}$, and $K_{4,B}$ while the proposed scheme uses N sets of those twelve keys. Therefore, the total number of the encryption keys in our scheme increases depending on the number of regions. Here, we consider the efficient key derivation scheme using hash chains. A hash chain implements a one-way hash function iteratively. By using hash chains, the different keys can be serially derived from single managed key $K_{0,R}$. Figure 7 shows the hierarchical key derivation using hash chains. We also introduce different constant numbers a_i (i = 1, 2, ..., N) and bitwise Ex-OR operations. The encryption keys in the proposed scheme are given as

$$K_{n+1,R} = H(K_{n,R} \oplus a_{n+1}), \quad n = 0, 1, ..., N-1, \tag{3}$$

$$K_{i,G} = H(K_{i,R}), \quad i = 1, 2, ..., N, \tag{4}$$

$$K_{i,B} = H(K_{i,G}), \quad i = 1, 2, ..., N, \tag{5}$$

where $H(\cdot)$ is a hash function. Owing to hash chains, it is quite difficult to compute key $K_{n,R}$ from $K_{n+1,R}$, and access control can be achieved. For instance, if a user receives key $K_{\gamma,R}$ (γ = 1, 2, ..., or N), he can generate $K_{\gamma,G}$, $K_{\gamma,B}$, and decrypt $R(\gamma)$. Additionally, the user derives keys $K_{\gamma+1,x}$, $K_{\gamma+2,x}$, ..., $K_{N,x}$ (x = R, G, B) and decrypts region $R(i)$, where $\gamma < i \leqq N$. The user, however, cannot access to any region $R(i)$, where $1 \leqq i < \gamma$.

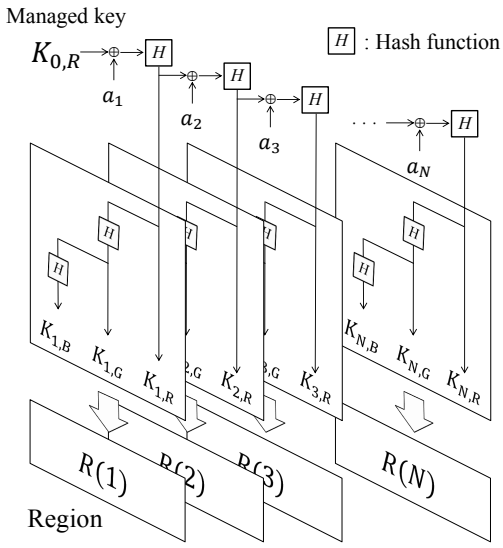


Fig. 7 Hierarchical key derivation.

IV. Experimental Results

We evaluate the encrypted images produced by the proposed algorithm from the aspects of the entropy, color distribution, and compression efficiency. The four kodak images [14] shown in Fig. 8 were used as test images. The size of the test images is 2,048 × 3,072 pixels. We divide each image into four same-sized regions and subdivide each region into multiple blocks with 32×32 pixels.

Figures 9 and 10 show the encrypted images produced by the proposed and the conventional [5] schemes. The invisibility of the encrypted images in the proposed scheme is analogous to that in the conventional scheme.

A. Entropy

We calculate the entropies of the original images and their encrypted images. The entropy $H(A)$ is defined as

$$H(A) = -\sum_{j=1}^{2^{24}} p(b_j)\log_2 p(b_j), \tag{6}$$

where A represents the finite set of 24-bit colors b_i (j = 1,2, ..., 2^{24}), that is , $A = \{b_1, b_2, ..., b_{2^{24}}\}$, and $p(b_j)$ is the occurrence probability of b_j. As shown in Table I, the entropy of each encrypted image in the proposed scheme is approximately equal to that in the conventional scheme.

B. Color Distribution

We compare the color distributions of the original images and their encrypted images, where the color components are transformed identically or independently. We convert the color space of each image from RGB to HSV to obtain the histogram based on hue and saturation. Figures 11 and 12 show the histograms of the single images and the combined images, where the x and y axes represent hue and saturation values, respectively. The encrypted images have wider distributions compared to the original images. Additionally, when the RGB components are transformed identically, the encrypted images of the combined image show the smoother distributions over the entire xy plane than the encrypted images of the single images. These results correspond to the entropies shown in Table I. Therefore, it is presumed that combining multiple images would be effective against JPSs in BPBE.

C. Compression Efficiency

Table II indicates the compression efficiency of the original images and their encrypted images. We compare the compression efficiency of JPEG-LS [15] among them. The compression efficiency is obtained by calculating the bitrates.

$$Bitrate(bpp) = \frac{Size\ of\ image\ file}{Number\ of\ pixels\ in\ image\ (M\times N)}, \tag{7}$$

where M and N are the vertical/horizontal sizes of the image. It is obvious that the proposed scheme can hold the same compression efficiency as that of the conventional scheme.

V. Conclusion

We proposed an adaptable BPBE scheme, which allows hierarchical decryption. The proposed scheme can retrieve the arbitrary regions of the encrypted image without decrypting other regions. We further discussed the invisibility of the encrypted images by comparing entropies and color distributions. It is more effective to apply our scheme to combined images for increasing the resilience against JPSs. The proposed scheme also maintains the compression efficiency of JPEG-LS relative to the conventional scheme. In addition, we proposed a superior key management scheme using hash chains.

References

[1] M. Kumar and A. Vaish, "An Efficient Encryption-then-Compression Technique for Encrypted Images Using SVD," *Digital Signal Processing*, vol.60, pp.81-89, 2017.

[2] W. Liu, W. Zeng, L. Dong, and Q. Yao, "Efficient Compression of Encrypted Grayscale Images," *IEEE Trans. Image Process.*, vol.19, no.4, pp.1097-1102, 2010.

[3] 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. Inf. Forensics Secur.*, vol.9, no.1, pp.39-50, 2014.

[4] K.G. Nimbokar, M.V. Sarode, and M.M. Ghonge, "A Survey Based on Designing an Efficient Image Encryption-then-Compression System," in *Proc. IJCA National Level Technical Conference X-PLORE 2014*, pp.6-8, 2014.

[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, 2015.

[6] O. Watanabe, A. Uchida, T. Fukuhara, and H. Kiya, "An Encryption-then-Compression System for JPEG 2000 Standard," in *Proc. IEEE ICASSP*, pp.1226-1230, 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.

(a) Image 1 (b) Image 2 (c) Image 3 (d) Image 4 (e) Combined image

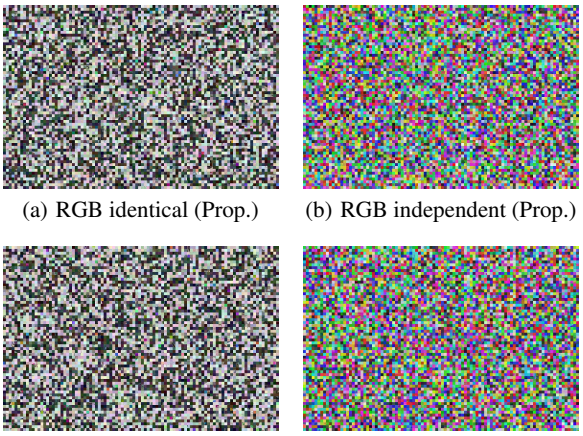
Fig. 8 Original images and combined image.

TABLE I Comparisons of entropies.

	Original	Proposed		Conventional [5]	
RGB	-	Identical	Independent	Identical	Independent
Image 1	14.17	17.44	21.75	17.44	21.75
Image 2	13.76	16.88	21.40	16.88	21.42
Image 3	14.82	17.75	21.51	17.75	21.52
Image 4	14.98	17.54	21.04	17.54	21.04
Combined	15.79	18.69	22.70	18.69	22.70

TABLE II Comparisons of compression efficiency.

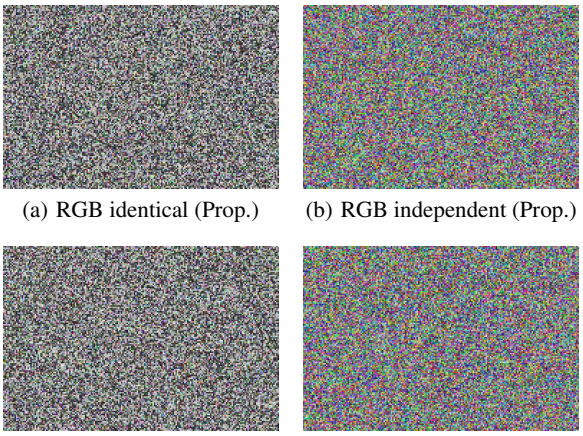
	Original		Proposed				Conventional [5]			
RGB	-		Identical		Independent		Identical		Independent	
-	Bitrate [bpp]	Compresssion ratio [%]	Bitrate [bpp]	Compresssion ratio [%]	Bitrate [bpp]	Compresssion ratio [%]	Bitrate [bpp]	Compresssion ratio [%]	Bitrate [bpp]	Compresssion ratio [%]
Image 1	11.31	52.88	11.67	51.37	11.68	51.33	11.67	51.37	11.68	51.33
Image 2	10.91	54.55	11.25	53.12	11.26	53.07	11.25	53.12	11.26	53.07
Image 3	9.52	60.32	9.90	58.75	9.91	58.71	9.90	58.75	9.90	58.71
Image 4	7.80	67.50	8.08	66.35	8.08	66.33	8.08	66.35	8.08	66.33
Combined	9.93	58.61	10.34	56.94	10.34	56.91	10.34	56.94	10.34	56.91



(a) RGB identical (Prop.) (b) RGB independent (Prop.)

(c) RGB identical (Conv. [5]) (d) RGB independent (Conv. [5])

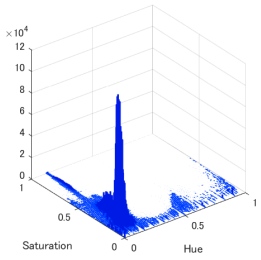
Fig. 9 Encrypted images of Image 1 shown in Fig. 8(a).



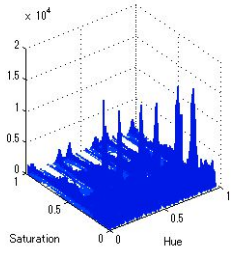
(a) RGB identical (Prop.) (b) RGB independent (Prop.)

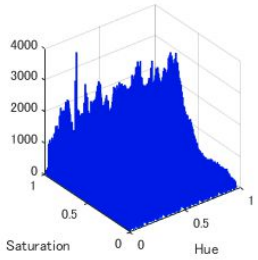
(c) RGB identical (Conv. [5]) (d) RGB independent (Conv. [5])

Fig. 10 Encrypted images of combined image shown in Fig. 8(e).



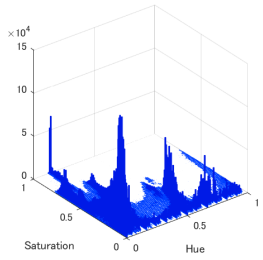
(a) Original image



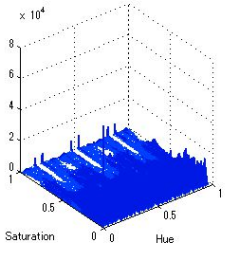


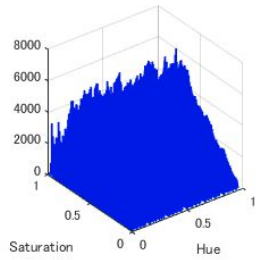
(b) Encrypted image (RGB identical) (c) Encrypted image (RGB independent)

Fig. 11 Comparing color distribution of single images. Original image is Image 1 shown in Fig. 8(a).



(a) Original image





(b) Encrypted image (RGB identical) (c) Encrypted image (RGB independent)

Fig. 12 Comparing color distribution of combined image. Original image is combined image shown in Fig. 8(e).

[8] S. Imaizumi and H. Kiya, "A Block-Permutation-Based Encryption Scheme with Independent Processing of RGB Components," *IEICE Trans. Inf. & Syst.*, vol.E101-D, no.12, 2018, *Accepted*.

[9] W. Sirichotedumrong, T. Chuman, S. Imaizumi, and H. Kiya, "Grayscale-Based Block Scrambling Image Encryption for Social Networking Services," in *Proc. IEEE ICME*, 2018.

[10] G. Paikin and A. Tal, "Solving Multiple Square Jigsaw Puzzles with Missing Pieces," in *Proc. IEEE CVPR*, pp.4832-4839, 2015.

[11] K. Son, D. Moreno, J. Hays, and D.B. Cooper, "Solving Small-piece Jigsaw Puzzles by Growing Consensus," in *Proc. IEEE CVPR*, pp.1193-1201, 2016.

[12] D. Sholomon, O.E. David, and N.S. Netanyahu, "An Automatic Solver for Very Large Jigsaw Puzzles Using Genetic Algorithms," *Genetic Programming and Evolvable Machines*, vol.17, no.3, pp.291-313, 2016.

[13] T. Chuman, K. Kurihara, and H. 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, 2018.

[14] [Online] Available: http://www.math.purdue.edu/~lucier/PHOTO_CD/D65_TIFF_IMAGES/

[15] M.J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I Lossless Image Compression Algorithm : Principles and Standardization into JPEG-LS," *IEEE Trans. Image Process.*, vol.9, no.8, pp.1309-1324, 2000.