PAPER *Special Section on Smart Multimedia & Communication Systems*

Cube-Based Encryption-then-Compression System for Video Sequences

Kosuke SHIMIZU[†a)], ***Student Member***, **Taizo SUZUKI**[††], ***and*** **Keisuke KAMEYAMA**[††], ***Members***

SUMMARY We propose the cube-based perceptual encryption (C-PE), which consists of cube scrambling, cube rotation, cube negative/positive transformation, and cube color component shuffling, and describe its application to the encryption-then-compression (ETC) system of Motion JPEG (MJPEG). Especially, cube rotation replaces the blocks in the original frames with ones in not only the other frames but also the depth-wise cube sides (spatiotemporal sides) unlike conventional block-based perceptual encryption (B-PE). Since it makes intra-block observation more difficult and prevents unauthorized decryption from only a single frame, it is more robust than B-PE against attack methods without any decryption key. However, because the encrypted frames including the blocks from the spatiotemporal sides affect the MJPEG compression performance slightly, we also devise a version of C-PE with no spatiotemporal sides (NSS-C-PE) that hardly affects compression performance. C-PE makes the encrypted video sequence robust against the only single frame-based algorithmic brute force (ABF) attack with only 21 cubes. The experimental results show the compression efficiency and encryption robustness of the C-PE/NSS-C-PE-based ETC system. C-PE-based ETC system shows mixed results depending on videos, whereas NSS-C-PE-based ETC system shows that the BD-PSNR can be suppressed to about −0.03 dB not depending on videos.

key words: *cube-based perceptual encryption, encryption-then-compression system, Motion JPEG*

1. Introduction

These days, large amounts of multimedia contents are compressed before being sent over bandwidth-constrained channels of communications networks. Contents posted in social networking services (SNSs) can be viewed by users without requiring any authorization other than signing in to the service; some such content, however, should be protected in view of privacy. One way of doing so is to encrypt the content before or after compressing it with the corresponding encryption key. In particular, the JPEG committee, which has studied the issue through its activity called JPEG Privacy and Security [1], has identified three kinds of JPEG-related encryption frameworks described below.

The compression-then-encryption (CTE) system [2]–[5] encrypts the content after compressing it (Fig. 1(a)). The sender **Alice** encrypts the content after compressing it, and the receiver **Bob** decrypts the content before decompressing it. The CTE system can also encrypt previously compressed content. It employs either mathematical or perceptual encryption. The mathematical encryption converts the input data into irregular number sequences, as in AES, RSA, and DES [6], [7]. The perceptual encryption is a format-compliant method that encrypts the content visibly. However, whichever encryption is selected, the CTE system cannot cope with SNSs that support re-encoding because the encrypted signal may not be decompressed without decryption.

The joint compression and encryption (JCE) system [8]–[12] encrypts a content while it is being compressed (Fig. 1(b)). Here, **Alice** encrypts the content during compression, and **Bob** decrypts it during decompression. To avoid destroying the standardized syntax of the codec, the JCE system uses perceptual encryption. It can encrypt content quickly and make it difficult to perform unauthorized decryption without decryption key, e.g., by using low-computational complexity and high-invisibility approaches that apply sign flips in the frequency domain [11], [12]. However, the JCE system is not easy to implement, and like the CTE system, it cannot cope with SNSs that support re-encoding.

The encryption-then-compression (ETC) system [13]–[16] encrypts the content before compressing it (Fig. 1(c)). **Alice** encrypts the content before compressing it, and **Bob** decrypts the content after decompressing it. The ETC system employs perceptual encryption so that existing compression frameworks such as JPEG can be used directly. Unlike CTE and JCE, it can cope with SNSs that support re-encoding. Therefore, this paper focuses on the ETC system.

The conventional ETC system for Motion JPEG (MJPEG) [13] employs block-based perceptual encryption (B-PE) consisting of four methods: block scrambling, block rotation/inversion, block negative/positive (nega-/posi-) transformation, and block color component shuffling. B-PE is applied to each frame independently, and its application to the ETC system can decompress and decrypt each frame in real time. However, since the original blocks remain in the encrypted frame, the encrypted frames may be decrypted by using attack methods without any decryption key such as the jigsaw-puzzle solver (JPS) attack [17].

To make the intra-block observation difficult and prevent unauthorized decryption from only a single frame, we propose cube-based perceptual encryption (C-PE), which consists of four methods: cube scrambling, cube rotation, cube nega-/posi-transformation, and cube color component shuffling. We aim to enhance the security even at the expense of real-time processing and complexity. Since cube rotation

Manuscript received February 2, 2018.
Manuscript revised June 24, 2018.

[†]The author is with the Department of Computer Science, University of Tsukuba, Tsukuba-shi, 305-8573 Japan.
[††]The authors are with the Faculty of Engineering, Information and Systems, University of Tsukuba, Tsukuba-shi, 305-8573 Japan.
a) E-mail: shimizu@adapt.cs.tsukuba.ac.jp

DOI: 10.1587/transfun.E101.A.1815

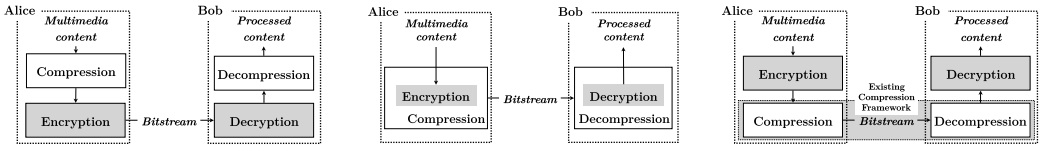


Fig. 1 Encryption and compression frameworks: (a) CTE system, (b) JCE system, and (c) ETC system.

makes intra-block observation more difficult and prevents unauthorized decryption from only a single frame, C-PE is robust against attacks without any decryption key, including the JPS attack, the cube-based JPS (CJPS) attack that we propose, the algorithmic brute force (ABF) attack, and the key-based brute force (KBF) attack. However, cube rotation affects the MJPEG compression performance slightly (unlike cube scrambling, cube nega-/posi-transformation, and cube color component shuffling) because the depth-wise cube sides (spatiotemporal sides) appear in the observed frame. As a remedy, we describe C-PE with no spatiotemporal sides (NSS-C-PE) that hardly affects compression performance. The results of experiments we conducted show the compression efficiency and encryption robustness of the C-PE/NSS-C-PE-based ETC system.

A preliminary version of this paper was presented in [18], where we discussed only a part of the C-PE. This paper describes the whole C-PE scheme and NSS-C-PE and their robustness against various attack methods.

2. Block-Based Encryption-then-Compression System

2.1 Block-Based Perceptual Encryption

The procedure of the conventional B-PE scheme is shown in Fig. 2(a). Each frame of the input video sequence is divided into blocks on which four methods are applied, as described in Sects. 2.1.1–2.1.4. When B-PE uses smaller blocks, visibility is further reduced. On the other hand, its application to the ETC system with MJPEG hardly affects compression performance when the block size is appropriately selected. A suitable block size for B-PE is $16k \times 16l$ ($^\forall k, l \in \mathbb{Z}_{>0}$). Moreover, the optimal block size, which reduces visibility as much as possible and does not affect MJPEG compression performance, is 16×16. This is because of that the MJPEG downsamples each frame from 16×16 blocks to 8×8 blocks in the chroma components and applies 2D discrete cosine transforms (DCTs) and quantization to the 8×8 downsampled blocks.

2.1.1 Block Scrambling

Block scrambling permutes sequentially numbered blocks in accordance with the number of blocks. It is clear that block scrambling does not affect MJPEG compression performance at all when a suitable block size, in our case $16k \times 16l$, is selected.

2.1.2 Block Rotation/Inversion

Block rotation/inversion rotates/inverts each block through a random angle and direction. Let $\mathcal{P}_i := \{(\mathcal{R}_i, \mathcal{G}_i, \mathcal{B}_i)\}$, $\mathcal{P}'_i$, and $\varepsilon_m(n)$ be the set of pixels and the RGB components in the ith block, the ith encrypted block, and the m-ary random number calculated in the nth block, respectively. The block rotation and block inversion are described as

$$\mathcal{P}'_i = \begin{cases} \mathcal{P}_i \circlearrowleft 0^\circ & (\varepsilon_4(i) = 0) \\ \mathcal{P}_i \circlearrowleft 90^\circ & (\varepsilon_4(i) = 1) \\ \mathcal{P}_i \circlearrowleft 180^\circ & (\varepsilon_4(i) = 2) \\ \mathcal{P}_i \circlearrowleft 270^\circ & (\varepsilon_4(i) = 3) \end{cases}, \tag{1}$$

where $\circlearrowleft$ means rotation, and

$$\mathcal{P}'_i = \begin{cases} \mathcal{P}_i \leftrightharpoons \textit{no inversion} & (\varepsilon_4(i) = 0) \\ \mathcal{P}_i \leftrightharpoons \textit{horizontally} & (\varepsilon_4(i) = 1) \\ \mathcal{P}_i \leftrightharpoons \textit{vertically} & (\varepsilon_4(i) = 2) \\ \mathcal{P}_i \leftrightharpoons \textit{diagonally} & (\varepsilon_4(i) = 3) \end{cases}, \tag{2}$$

where $\leftrightharpoons$ means inversion. The block rotation/inversion affects the MJPEG compression performance only slightly when a suitable block size, i.e., $16k \times 16l$, is selected because the quantization table is not symmetric.

2.1.3 Block Negative/Positive Transformation

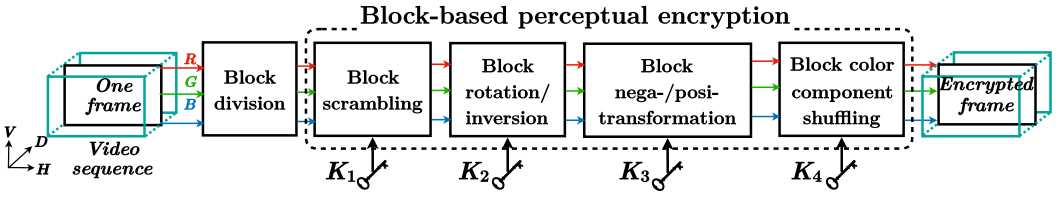
The block nega-/posi-transformation inverts the colors of the blocks randomly. A nega-/posi-transformation in the ith block is described as

$$\mathcal{P}'_i = \begin{cases} \mathcal{P}_i & (\varepsilon_2(i) = 0) \\ \{255\} - \mathcal{P}_i & (\varepsilon_2(i) = 1) \end{cases}. \tag{3}$$

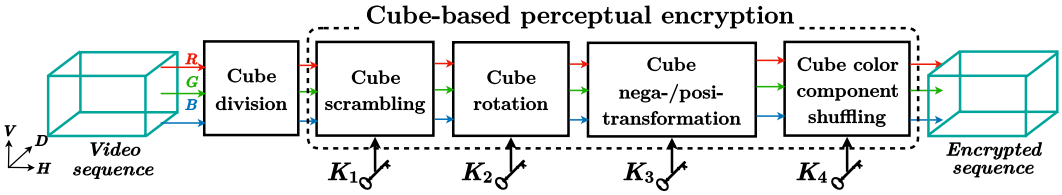
Like block rotation/inversion, the block nega-/posi-transformation affects MJPEG compression performance only slightly when a suitable block size, i.e., $16k \times 16l$, is selected, because the signal characteristics in the nega-/posi-transformed block change only a little.

2.1.4 Block Color Component Shuffling

Block color component shuffling exchanges the order of the color components in each block. The block color component shuffling is described as



(a)



(b)

Fig. 2 Block-/cube-based perceptual encryption: (a) block-based and (b) cube-based.

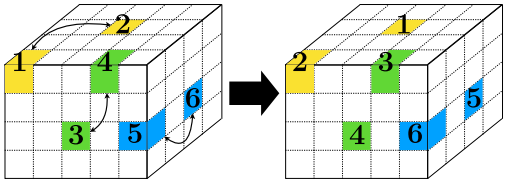


Fig. 3 Cube scrambling.

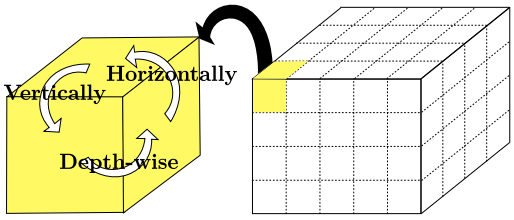


Fig. 4 Cube rotation.

$$\mathcal{P}'_i = \begin{cases} \{(\mathcal{R}_i, \mathcal{G}_i, \mathcal{B}_i)\} & (\varepsilon_6(i) = 0) \\ \{(\mathcal{R}_i, \mathcal{B}_i, \mathcal{G}_i)\} & (\varepsilon_6(i) = 1) \\ \{(\mathcal{G}_i, \mathcal{R}_i, \mathcal{B}_i)\} & (\varepsilon_6(i) = 2) \\ \{(\mathcal{G}_i, \mathcal{B}_i, \mathcal{R}_i)\} & (\varepsilon_6(i) = 3) \\ \{(\mathcal{B}_i, \mathcal{R}_i, \mathcal{G}_i)\} & (\varepsilon_6(i) = 4) \\ \{(\mathcal{B}_i, \mathcal{G}_i, \mathcal{R}_i)\} & (\varepsilon_6(i) = 5) \end{cases}. \quad (4)$$

Like block rotation/inversion and block nega-/posi-transformation, block color component shuffling affects the JPEG compression performance only slightly when a suitable block size, i.e., $16k \times 16l$, is selected, because the YUV components obtained from the shuffled RGB components are somewhat different from the original YUV components.

2.2 Advantages and Disadvantages

The advantage of B-PE is that the encrypted frames maintain almost the same MJPEG compression performance as the unencrypted ones when a suitable block size, i.e., $16k \times 16l$, is selected. Furthermore, the encrypted frames can be decrypted in order and in real time.

However, real-time decryption faces a risk. Each block of the encrypted frames can be observed in each frame by anyone. Therefore, each encrypted frame may be decrypted by applying methods such as the JPS attack without any decryption key. To prevent unauthorized decryption, the blocks in the encrypted frames should be moved from their original frames to other frames.

Algorithm 1: Cube scrambling

Input: Video sequence V_{in}, cube depth ℓ ($\ell \in \mathbb{Z}_{>0}$)
Output: Encrypted V_{in}
Initialize N, N_D, N_V, N_H, N_S, and $\mathcal{A}$ using (5)
for $i = 0$ *to* $\lfloor (N-1)/2 \rfloor$ **do**
  Compute (D_{2i}, V_{2i}, H_{2i}) using (6)
  Compute $(D_{2i+1}, V_{2i+1}, H_{2i+1})$ using (6)
  //Exchanging two cubes
  $\ell \times \ell \times \ell$ cube from $V_{in}(D_{2i}, V_{2i}, H_{2i}) \leftrightarrows$ $\ell \times \ell \times \ell$ cube from $V_{in}(D_{2i+1}, V_{2i+1}, H_{2i+1})$

3. Cube-Based Encryption-then-Compression System

3.1 Cube-Based Perceptual Encryption

The procedure of the C-PE is shown in Fig. 2(b). Frames of the input video sequence are bundled as a large cuboid and divided into the small cubes to apply cube scrambling, cube rotation, cube nega-/posi-transformation, and cube color component shuffling.

3.1.1 Cube Scrambling

The cube scrambling is shown in Fig. 3. It permutes the cubes randomly in the video sequence. When the cubes are

Algorithm 2: Cube rotation

Input: Video sequence V_{in}, cube depth ℓ ($\ell \in \mathbb{Z}_{>0}$)
Output: Encrypted V_{in}
Initialize N, N_D, N_V, N_H, N_S, and $\mathcal{A}$ using (5)
for $i = 0$ *to* $N - 1$ **do**
  Compute (D_i, V_i, H_i) using (6)
  $\vartheta \leftarrow \varepsilon_4(i)$ // 0, 1, 2, or 3
  //Depth-wise rotation
  Rotate $\ell \times \ell \times \ell$ cube from $V_{in}(D_i, V_i, H_i)$ by $(90 \times \vartheta)^\circ$ depth-wise
  $\vartheta \leftarrow \varepsilon_4(i)$
  //Vertical rotation
  Rotate $\ell \times \ell \times \ell$ cube from $V_{in}(D_i, V_i, H_i)$ by $(90 \times \vartheta)^\circ$ vertically
  $\vartheta \leftarrow \varepsilon_4(i)$
  //Horizontal rotation
  Rotate $\ell \times \ell \times \ell$ cube from $V_{in}(D_i, V_i, H_i)$ by $(90 \times \vartheta)^\circ$ horizontally

scrambled, the blocks in each original frame are moved to other frames. To return the encrypted blocks to the original positions, they must be returned from other frames, unlike in the conventional B-PE.

The cube scrambling algorithm is described in Algorithm 1. Here, D, H, W, and ℓ are the depth, height, width of the V_{in}, and the cube depth ℓ ($\ell \in \mathbb{Z}_{>0}$), respectively; N_D, N_V, N_H, N, and N_S are calculated as

$$\begin{aligned} N_D &= \left\lfloor \frac{D}{\ell} \right\rfloor,\ N_V = \left\lfloor \frac{H}{\ell} \right\rfloor,\ N_H = \left\lfloor \frac{W}{\ell} \right\rfloor, \\ N &= N_D \times N_V \times N_H,\ N_S = N_V \times N_H, \end{aligned} \tag{5}$$

where they are the depth-wise cube number, vertical cube number, horizontal cube number, total cube number, and cube number per $N_D = 1$. $\mathcal{A} := \{a_0, a_1, \cdots, a_{N-1} | a_i \in [0, N-1] \subset \mathbb{Z}\}$ ($i = 0, 1, \cdots, N-1$) is the shuffled number set in which the elements differ from each other and is generated from a pseudo random number generator (PRNG) [19] initialized with a seed based on the encryption key. While scanning $\mathcal{A}$, two cube start points (D_{2i}, V_{2i}, H_{2i}) and $(D_{2i+1}, V_{2i+1}, H_{2i+1})$ are calculated as

$$\begin{aligned} D_t &= \lfloor a_t / N_S \rfloor \times \ell, \\ V_t &= \lfloor (a_t \% N_S) / N_H \rfloor \times \ell, \\ H_t &= (a_t \% N_S) \% N_H \times \ell, \end{aligned} \tag{6}$$

where the % means the modulo operator. The two $\ell \times \ell \times \ell$ cubes extracted from $V_{in}(D_{2i}, V_{2i}, H_{2i})$ and $V_{in}(D_{2i+1}, V_{2i+1}, H_{2i+1})$ are iteratively exchanged by the operator $\leftrightarrows$.

3.1.2 Cube Rotation

The cube rotation method is shown in Fig. 4. It rotates the cubes through a random angle and direction. When the cubes are rotated through particular angles, the spatiotemporal sides are observable. Thus, the textures in the original blocks are exchanged with different ones. If the spatiotemporal sides appear in the observed frames, decryption with only a single frame clearly becomes impossible and the spatiotemporal redundancies are compressed by the MJPEG framework instead of intra-frame redundancies like in the other video compression standards: MPEG1, MPEG2, H.264/AVC, and H.265/HEVC. However, the spatiotemporal sides are unexpected textures that JPEG usually does not process so that they may greatly affect the compression performance of MJPEG. We will investigate this effect in Sect. 4.

Algorithm 2 is for cube rotation. Here, N, N_D, N_V, N_H, N_S, and $\mathcal{A}$ are initialized as before. The cube start point (D_i, V_i, H_i) is iteratively calculated while scanning $\mathcal{A}$. Randomly generated ϑ determines the rotation angle of the ith cube. For example, if the three ϑs are 3, 1, and 2, the ith cube extracted from (D_i, V_i, H_i) is rotated by 270° depth-wise, 90° vertically, and 180° horizontally. The above rotations are iterated. The vertical and depth-wise cube rotations into 90° and 270° mix the blocks in the spatiotemporal sides into the observed frames unlike B-PE. Also, note that a combination of the depth-wise cube rotation by 180° (0°) and the vertical cube rotation by 0° (180°) represents a combination of the block inversion and (spatiotemporal) block scrambling. Therefore, we have not proposed "cube inversion (an extension of block inversion)" specifically.

3.1.3 Cube Negative/Positive Transformation

The cube nega-/posi-transformation inverts the color values in the cubes randomly. Let $C_i := \bigcup_{t=0}^{\ell} \mathcal{P}_{it} = \bigcup_{t=0}^{\ell} \{(\mathcal{R}_{it}, \mathcal{G}_{it}, \mathcal{B}_{it})\}$ and C'_i be the ith cube including ℓ blocks and the ith encrypted cube, respectively. The transformation is described as

$$C'_i = \begin{cases} \bigcup_{t=0}^{\ell} (\{255\} - \mathcal{P}_{it}) & (\varepsilon_2(i) = 0) \\ \bigcup_{t=0}^{\ell} \mathcal{P}_{it} & (\varepsilon_2(i) = 1) \end{cases}. \tag{7}$$

3.1.4 Cube Color Component Shuffling

The cube color component shuffling permutes the color components of the cubes randomly. The transformation is described as

$$C'_i = \begin{cases} \bigcup_{t=0}^{\ell} \{(\mathcal{R}_{it}, \mathcal{G}_{it}, \mathcal{B}_{it})\} & (\varepsilon_6(i) = 0) \\ \bigcup_{t=0}^{\ell} \{(\mathcal{R}_{it}, \mathcal{B}_{it}, \mathcal{G}_{it})\} & (\varepsilon_6(i) = 1) \\ \bigcup_{t=0}^{\ell} \{(\mathcal{G}_{it}, \mathcal{R}_{it}, \mathcal{B}_{it})\} & (\varepsilon_6(i) = 2) \\ \bigcup_{t=0}^{\ell} \{(\mathcal{G}_{it}, \mathcal{B}_{it}, \mathcal{R}_{it})\} & (\varepsilon_6(i) = 3) \\ \bigcup_{t=0}^{\ell} \{(\mathcal{B}_{it}, \mathcal{R}_{it}, \mathcal{G}_{it})\} & (\varepsilon_6(i) = 4) \\ \bigcup_{t=0}^{\ell} \{(\mathcal{B}_{it}, \mathcal{G}_{it}, \mathcal{R}_{it})\} & (\varepsilon_6(i) = 5) \end{cases}. \tag{8}$$

3.2 Robustness Analysis

C-PE prevents unauthorized decryption from only a single

frame because of original blocks exchanged with different ones, as described in Sect. 3.1.2. However, other attacks which try to decrypt the encrypted video sequence without any decryption key may be conducted. This subsection investigates the robustness of C-PE against these sorts of attacks. We can consider that the C-PE is robust against them if their numbers of total brute force attacks are greater than 2^{256}, which is the number of brute force attack against the one-way hash function SHA-256 [20] and is a number that makes decryption of the encrypted content impossible in real time.

3.2.1 Jigsaw-Puzzle Solver Attack and Cube-Based Jigsaw-Puzzle Solver Attack

Chuman et al. have already presented that B-PE is robust enough against JPS attacks if the appropriate sizes of blocks and selection of B-PE methods can prevent JPS attacks [17]. Therefore, we can easily find that our C-PE, which includes all B-PE methods and mixes the blocks in the other frames and the spatiotemporal sides into the observed frames, has sufficient robustness against JPS attacks.

Also, we present a CJPS attack as a new JPS attack for C-PE. The CJPS attack chooses cubes of the encrypted video sequence and tries to assemble them correctly. This attack against C-PE corresponds to matching two cube sides after concatenation, rotation, color inversion, and shuffling of the color component order of the cubes. The number of CJPS attacks is calculated as follows. The number of sides per cube is 6. The number of color inversions and color component order shufflings is $2 \times 6 = 12$ per side. Namely, a sufficient number to match each cube side with the corresponding one is $6 \times 12 = 72$. If this operation is conducted on all N cubes, totally $\Sigma_{t=1}^{N}(2+4t)\times(24\times12)^{N-t}$ matches are required. Therefore, the CJPS attack cannot decrypt the content encrypted by C-PE in real time when the total number of cubes N is 33 or more.

3.2.2 Algorithmic Brute Force Attack

The ABF attack decrypts the encrypted video sequence by applying all possible candidates of this algorithm.

The number of ABF attacks against cube scrambling is calculated as follows. The probability of a cube being moved back to the correct place is $\frac{1}{N}$. The joint probability of another cube also being moved back to the correct place is $\frac{1}{N(N-1)}$. Accordingly, the joint probability of all cubes being moved back to the correct places is $\prod_{t=1}^{N}\frac{1}{t(N-t)} = \frac{1}{N!}$. Consequently, the number of the ABF attacks against cube scrambling is

$$N_{CS} = N! = 2^{\log_2 \prod_{i=1}^{N} i} = 2^{\sum_{i=1}^{N}\log_2 i}. \tag{9}$$

The number of ABF attacks against cube rotation is calculated as follows. The rotation directions are depth-wise, vertically, and horizontally. The rotation angle is $0°, 90°, 180°$, or $270°$. Thus, the probability with which the directions and angles of N cubes are correctly returned is $(1/4^3)^N = 1/64^N$. Consequently, the number of ABF attacks against the cube rotation is

$$N_{CR} = 64^N = 2^{6N}. \tag{10}$$

The number of ABF attacks against cube nega-/posi-transformation is calculated as follows. The probability with which the colors of N cubes are correctly inverted is $1/2^N$. Consequently, the number of ABF attacks against the cube nega-/posi-transformation is

$$N_{CN} = 2^N. \tag{11}$$

The number of ABF attacks of cube color component shuffling is calculated as follows. The probability with which the colors of N cubes are correctly shuffled is $1/6^N$. Consequently, the number of ABF attacks against the cube nega-/posi-transformation is

$$N_{CC} = 6^N = 2^{N\log_2 6}. \tag{12}$$

The total number of ABF attacks against all methods of C-PE is

$$N_C = N_{CS}N_{CR}N_{CN}N_{CC} = 2^{7N+N\log_2 6+\sum_{i=1}^{N}\log_2 i}. \tag{13}$$

Therefore, we can consider that the ABF attack cannot decrypt the C-PE encrypted content in real time when the total number of cubes N is 21 or more.

3.2.3 Key-Based Brute Force Attack

The KBF attack decrypts the encrypted content by applying all possible keys.

The secrecy of the key of C-PE depends on the PRNG. The PRNG generates the random numbers from a primitive seed δ by performing a recursive calculation. However, δ is 4 bytes, which is too short to prevent attacks on it, so that it is inapplicable as an encryption key. Thus, the key should be long data such as the SHA-256 hash digest. If the SHA-256 hash digest is used as the parent key Γ, each Γ_t $(t \in \mathbb{Z}_{\geq 0})$ for each C-PE method can be generated [13], [21] as

$$\begin{aligned} \Gamma_0 &= \Gamma, \\ \Gamma_{t+1} &= \mathcal{S}(\Gamma_t), \end{aligned} \tag{14}$$

where $\mathcal{S}(\cdot)$ is a hash function. The seeds δs are recursively extracted from Γ_{t+1} and applied to the encryption methods and the encrypted regions. The regions are selected in any way: a key, which $\mathcal{S}(\cdot)$ has been applied, can be applied to a C-PE method for whole video sequence or for each region bundling 16 frames of video sequence. Although having only Γ allows one to decrypt the content with or without the correct key, the computational time to find Γ has been believed to be very long, because the preimage of the SHA-2 family cannot be calculated in real time [22].

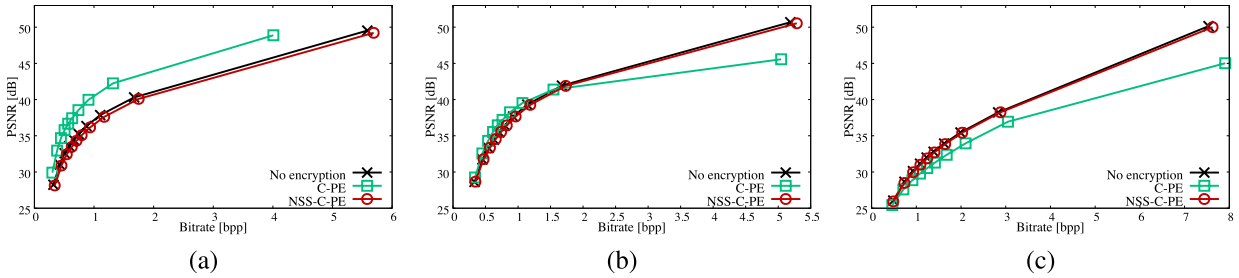


Fig. 5 R-D curve (average of 256 frames): (a) *Akiyo*, (b) *Bowing*, and (c) *Coastguard*.

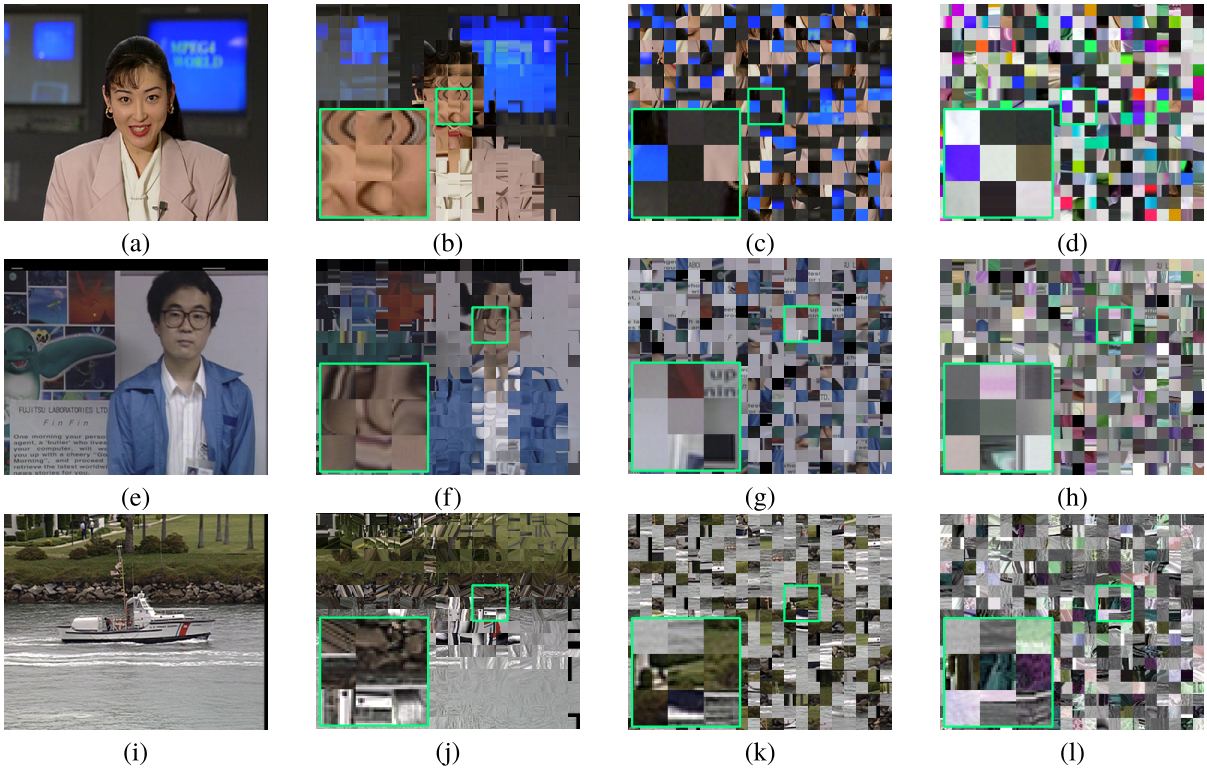


Fig. 6 Frames in which a sequence was encrypted by $16 \times 16 \times 16$ cubes: (top-to-bottom) *Akiyo's* 200th frame, *Bowing*'s 160th frame, and *Coastguard*'s 140th frame, (left-to-right) original, only cube rotation, only cube scrambling, and all C-PEs.

Table 1 Test video sequences.

Input video sequence	*Akiyo*	*Bowing*	*Coastguard*
Moving area	small	medium	large
Stopping area	large	medium	small
Size & color depth	$288 \times 352 \times 256$, 8-bit RGB		

Table 2 BD-PSNR [dB] of each encryption method applied to *Akiyo*, *Bowing*, and *Coastguard*.

Method	*Akiyo*	*Bowing*	*Coastguard*
Cube scrambling	−0.25	−0.20	−0.11
Cube rotation (C-PE)	**3.66**	−0.17	**−2.01**
Cube rotation (NSS-C-PE)	−0.01	−0.03	−0.02
Cube nega-/posi-transformation	−0.15	−0.13	−0.05
Cube color component shuffling	−0.44	−0.17	−0.09

4. Experiments

First, we describe the experimental conditions and procedures. Then, we describe the compression efficiency and encryption robustness of video sequences encrypted by C-PE and NSS-C-PE. NSS-C-PE is the variant of C-PE that does not show the spatiotemporal sides by cube rotation; i.e., it does not conduct depth-wise and vertical rotations by 90° and 270°.

4.1 Experimental Conditions and Procedure

The test video sequences [23] are shown in Table 1 and Fig. 6(a), (e), (i). The purpose of the experiments was to measure the compression efficiencies of encrypted video sequences whose movements depicted therein were different from each other. The size of the cube in the experiments was set as $16 \times 16 \times 16$ in accordance with [13].

The common procedure was as follows.

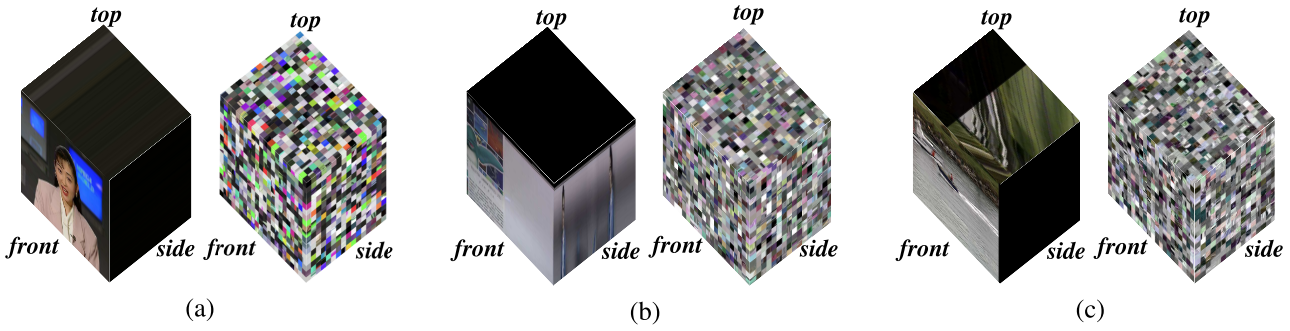


Fig. 7 Results of encryption: (a) original *Akiyo* and encrypted with C-PE, (b) original *Bowing* and encrypted with C-PE, (c) original *Coastguard* and encrypted with C-PE.

1. Encrypt the input frames.
2. Compress the encrypted frames with *libjpeg-turbo* [24], whose compression qualities are $Q := \{10, 20, \cdots, 100\}$.
3. Calculate the mean bitrates of the compressed frames.
4. Decompress the compressed frames.
5. Decrypt the decompressed frames.
6. Calculate the mean Bjøntegaard delta PSNRs (BD-PSNRs), which correspond to the average PSNR difference [dB] for the same bit rate, between the input frames and the decrypted frames.

4.2 Results of C-PE

Here, we compare the MJPEG compression performances of the encrypted video sequences encrypted with each C-PE method. The BD-PSNRs of each C-PE method for *Akiyo*, *Bowing*, and *Coastguard* are listed in Table 2. The cube rotation affected the MJPEG compression performance, especially for *Akiyo* and *Coastguard*, unlike the cube scrambling, cube nega-/posi-transformation, or cube color component shuffling.

The results of all C-PE methods are shown as green lines in Fig. 5. The rate-distortion (R-D) curves denote comparisons of compression performance between video sequences encrypted with C-PE and unencrypted ones (black lines). The compression performance of the encrypted *Akiyo* was better than that of the unencrypted one. The compression performance of the encrypted *Bowing* was almost the same as that of the unencrypted one[†]. The compression performance of encrypted *Coastguard* fell below that of the unencrypted one. We consider that the performance of C-PE-based MJPEG compression depends on the video.

The visibilities of the encrypted frames are shown in Fig. 6. When only cube rotation was applied, the spatiotemporal sides appeared in some blocks of the frame (Fig. 6(b), (f), (j)). When only cube scrambling was applied, the original blocks in the frame were hidden by exchanging them with other blocks of the other frames (Fig. 6(c), (g), (k)). When all encryption methods were applied, the encrypted blocks differed from the original blocks (Fig. 6(d), (h), (l)). The whole encrypted results of *Akiyo*, *Bowing*, and *Coastguard* are shown in Fig. 7. As can be seen, many blocks were moved from the observed frames to the spatiotemporal sides. Therefore, it is clear that the encrypted frame cannot be without any decryption key decrypted from only itself.

[†]The difference at the highest two points (whose quality is 100) is not important because humans have difficulty perceiving differences in images with PSNRs greater than 40 dB.

4.3 Results of NSS-C-PE

The compression performances of NSS-C-PE are shown as red lines in Fig. 5. The NSS-C-PE hardly affected the MJPEG compression performance because the cube rotation did not put the spatiotemporal sides in any frames (Table 2 Cube rotation (NSS-C-PE)).

Regarding the encryption robustness of NSS-C-PE, the number of ABF attacks against cube rotation with no spatiotemporal sides becomes 2^{4N}. Thus, the total number becomes $N'_C > 2^{5N+N\log_2 6+\sum_{i=1}^{N}\log_2 i}$, and a sufficient cube number N is 24. Therefore, it is considered that NSS-C-PE is robust against the ABF attack by dividing the input video sequence into 24 or more cubes. On the other hand, the robustness against the CJPS attack is the same as in the case of the full C-PE because no observer can distinguish whether it is full C-PE or NSS-C-PE. The robustness against the KBF attack appears to be the same as in the case of the full C-PE. However, it may be less than that of the full C-PE because the original textures appear in each block.

5. Conclusion

We proposed cube-based perceptual encryption, which consists of cube scrambling, cube rotation, cube nega-/posi-transformation, and cube color component shuffling, and described its application to the ETC system with MJPEG. Since C-PE makes the intra-block observation difficult and prevents unauthorized decryption from only a single frame, it is more robust than the conventional B-PE against attack methods without any decryption key. Although the cube rotation affects the MJPEG compression performance slightly, we proposed a variant, NSS-C-PE, that hardly affects compression performance. Experiments showed the compression efficiency and encryption robustness of the C-PE/NSS-

C-PE-based ETC system. As future works, the C-PE and NSS-C-PE will be applied with some block-based MJPEG-like compression standard with no motion compensations.

Acknowledgments

This work was supported by a JSPS Grant-in-Aid for Young Scientists, Grant Number 16K18100.

References

[1] "JPEG Privacy & Security Abstract and Executive Summary," https://jpeg.org/items/20150910_privacy_security_summary.html
[2] N.G. Bourbakis, "Image data compression-encryption using G-scan patterns," Proc. ICSMC'97, pp.1117–1120, Orlando, FL, Oct. 1997.
[3] H. Cheng and X. Li, "Partial encryption of compressed images and videos," IEEE Trans. Signal Process., vol.48, no.8, pp.2439–2451, Aug. 2000.
[4] M. Ito, N. Ohnishi, A. Alfalou, and A. Mansour, "New image encryption and compression method based on independent component analysis," Proc. ICTTA'08, pp.1–6, Damascus, Syria, April 2008.
[5] D. Maheswari and V. Radha, "Secure layer based compound image compression using XML compression," Proc. of ICCIC'10, pp.494–498, Coimbatore, India, Dec. 2010.
[6] J. Daemen and V. Rijmen, The Design of Rijndael AES — The Advanced Encryption Standard, Springer, 2002.
[7] Advanced Encryption Standard (AES), FIPS Pub. 197, Nov. 2001.
[8] W. Zeng and S. Lei, "Efficient frequency domain selective scrambling of digital video," IEEE Trans. Multimedia, vol.5, no.1, pp.118–129, April 2003.
[9] T. Lei, "Methods for encrypting and decrypting MPEG video data efficiently," Proc. ACMMM'96, pp.219–229, New York, NY, USA, Nov. 1996.
[10] C. Shi, S. Wang, and B. Bhargava, "MPEG video encryption in real-time using secret key cryptography," Proc. PDPTA'99, pp.1–7, Las Vegas, NV, USA, June-July 1999.
[11] B. Zeng, A. Yeung, S. Kei, S. Zhu, and M. Gabbouj, "Perceptual encryption of H.264 videos: Embedding sign-flips into the integer-based transforms," IEEE Trans. Inf. Forensics Secur., vol.9, no.2, pp.309–320, Feb. 2014.
[12] F. Dufaux and T. Ebrahimi, "Scrambling for privacy protection in video surveillance systems," IEEE Trans. Circuits Syst. Video Technol., vol.18, no.8, pp.1168–1174, Aug. 2008.
[13] 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, Nov. 2015.
[14] A. Kingston, S. Colosimo, P. Campisi, and F. Autrusseau, "Lossless image compression and selective encryption using a discrete Radon transform," Proc. ICIP'07, pp.465–468, San Antonio, TX, Sept. 2007.
[15] F. Ahmed, M.Y. Siyal, and V.U. Abbas, "A perceptually scalable and JPEG compression tolerant image encryption scheme," Proc. PRIVT'10, pp.232–238, Singapore, Nov. 2010.
[16] O. Watanabe, T. Fukuharai, and H. Kiya, "A perceptual encryption scheme for Motion JPEG 2000 standard," Proc. ISCIT'15, pp.125–128, Nara, Japan, Oct. 2015.
[17] 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. & Syst., vol.E101-D, no.1, pp.37–44, Jan. 2018.
[18] K. Shimizu and T. Suzuki, "Cube-based encryption connected prior to Motion JPEG standard," Proc. APSIPA ASC 2017, pp.1811–1814, Kuala Lumpur, Malaysia, Dec. 2017.
[19] M. Matsumoto and T. Nishimura, "Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator," ACM Trans. Model. Comput. Simul., vol.8, no.1, pp.3–30, Jan. 1998.
[20] Secure Hash Standard (SHS), FIPS PUB 180-4, Aug. 2015.
[21] M. Fujiyoshi, S. Imaizumi, and H. Kiya, "Encryption of composite multimedia contents for access control," IEICE Trans. Fundamentals, vol.E90-A, no.3, pp.590–596, March 2007.
[22] K. Dmitry, R. Christian, and S. Alexandra, Bicliques for Preimages: Attacks on Skein-512 and the SHA-2 Family, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.
[23] "Xiph.org Video Test Media [derf's collection]," https://media.xiph.org/video/derf/
[24] "JPEG software," https://jpeg.org/jpeg/software.html

Kosuke Shimizu graduated from the Computer Science Program of the Advanced Course, Tokyo Metropolitan College of Industrial Technology (TMCIT), Japan, and received the B.E. degree from the National Institution for Academic Degrees and University Evaluation, Japan, in 2017. In 2017, he joined a master's course in the Department of Computer Science, University of Tsukuba, Japan. His current research interest is image and video processing.

Taizo Suzuki received the B.E., M.E., and Ph.D. degrees in electrical engineering from Keio University, Japan, in 2004, 2006 and 2010, respectively. From 2006 to 2008, he was with Toppan Printing Co., Ltd., Japan. From 2008 to 2011, he was a Research Associate of the Global Center of Excellence (G-COE) at Keio University, Japan. From 2010 to 2011, he was a Research Fellow of the Japan Society for the Promotion of Science (JSPS) and a Visiting Scholar at the Video Processing Group, the University of California, San Diego, CA. From 2011 to 2012, he was an Assistant Professor in the Department of Electrical and Electronic Engineering, College of Engineering, Nihon University, Japan. Since 2012, he has been an Assistant Professor in the Faculty of Engineering, Information and Systems, University of Tsukuba, Japan. His current research interests include signal processing and filter banks/wavelets for image and video. Since 2017, he has been an Associate Editor of IEICE Trans. Fundamentals.

Keisuke Kameyama received the B.E., M.E., and Dr. Eng. degrees from Tokyo Institute of Technology. He is currently a Professor at the Faculty of Engineering, Information and Systems, University of Tsukuba. His research interests include pattern recognition, signal processing, and adaptive information processing methods. Dr. Kameyama is a member of IEICE, JNNS and IEEE.