Hierarchical image-scrambling method with scramble-level controllability for privacy protection

Toshiya Honda, Yuma Murakami, Yuki Yanagihara, Takeshi Kumaki, and Takeshi Fujino
Department of Electronic and Computer Engineering, Ritsumeikan University
1-1-1, Noji-Higashi, Kusatsu, Shiga, 525-8577, Japan
Telephone: +81-77-561-5150
Email: { ri008085@ed, ri009071@ed, ri0007hh@ed, kumaki@fc, fujino@se } .ritsumei.ac.jp

Abstract— **Privacy-protecting technology is essential in this surveillance society. The applications of cameras widely vary, e.g., crime surveillance, monitoring of an environment, and marketing. Furthermore, the scale of surveillance systems is predicted to become more diverse (from home area networks to wide area networks) due to the decrease in size and price of cameras. Therefore, a simple privacy protection system that does not require central servers or large databases is needed. Scrambling private information in a captured image can be a solution to simplifying a system. We propose an image-scrambling method for bitmap and JPEG formatted images to private information. Our method enables access control by providing keys to authorized individuals. They cannot view private information that they do not have permission to access. The image's format is retained; therefore, no special viewer is necessary in display-only console. Experimental results suggest that scramble level can be controlled linearly by using parameters (three for JPEG formatted image, and one for bitmap image). We also developed a demo system for this method and confirm that this method can be applied to embedded systems such as those equipped with surveillance cameras.**

I. Introduction

Cameras are now widely used for a variety of applications, e.g., crime surveillance, monitoring of an environment, and marketing. Since the size of the camera sensor and power consumption is decreasing, some researchers focused on applying camera sensors to wireless multimedia sensor networks (WMSNs) [1], [2].

However, invasion of privacy has become a serious social problem, especially due to the use of surveillance cameras. The Surveillance Studies Network in the United Kingdom reported on the surveillance society [3]. This report argues that a large number of our daily activities are under surveillance, and surveillance has both benefits and risks. Therefore, surveillance requires "regulation", e.g., applying rules or setting limits and controls. As a result, some surveillance devices have functions to limit accessibility to information. For example, current surveillance cameras on the market have an un-restorable privacy protection function. It conceals private information by pixel overwriting, such as filling or creating mosaics, to the image. It can change concealing method according to the target object. However, it is not possible to restore the original image with this method. In criminal investigation, for example, such a method could render evidence captured as an image useless.

Methods have been proposed as measures to cope with invasion of privacy. IBM Smart Surveillance System, previously called the PeopleVision project, is a middleware system equipped with a a surveillance camera system with privacy protection [4]. Senior reported a part of a project for developing a technology to protect privacy in video systems [5]. It minimizes intrusion into individuals' privacy by transforming private information. It also encrypts the video stream that is transformed (e.g., removed, replaced, or reduced in resolution) according to the system policy. However, the transforming process is un-restorable; therefore, the transformed information cannot be used as original information.

Such large systems are not suitable for small-scale surveillance (e.g., home networks and WMSNs), because they require database servers for storing the analyzed information , and complex subsystems. In such systems, it is reasonable to scramble an image in the camera, and distribute information of scrambled area and parameters within the scrambled image to save system costs. Therefore, we should apply image privacy-protecting technology to cameras and consoles. Methods have been proposed for scrambling private information and distribution. Yabuta et al. proposed a restorable masking method for JPEG (Joint Photographic Experts Group) formatted images [6]. This method converts private information to JPEG format and embeds it to a masked (e.g., mosaic, permeated, or filled) image by watermarking. There is a limitation to the data size of embedding information due to this watermarking. Fujii et al. proposed an image-scrambling method for content distribution [7]. This method inputs a JPEG image or MPEG (Moving Picture Experts Group) video stream. They have coded by entropy encoding; therefore, they have many pairs of Huffman-code and optional bit. This method encrypts these optional bits to enable scrambling. This method can change the quality of the masked area by using parameters and density of scrambling. However, especially in high-rate compressed data, the length of optional bits will be zero or a few bits. Therefore, an attacker may be able to restore the original image by brute-force attacking of the coefficients.

Our goal of this study is to develop a light-weight (applicable to an embedded system) and secure (regardless of contents in an image) image scrambling method for privacy protection. In this paper, we propose a restorable image-scrambling method for bitmap and JPEG formatted images

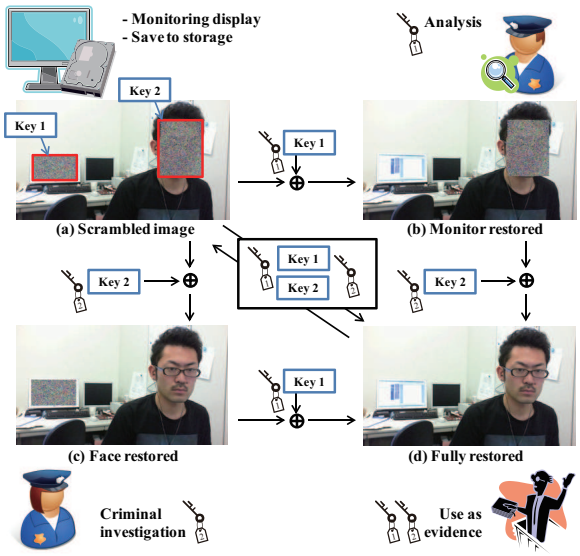


Fig. 1. Application example: indoor surveillance system

for protecting privacy. This method has a simple structure that consists of a cipher subsystem, object detector, and format conversion module (e.g., JPEG compressor). It can scramble an image hierarchically by applying different keys and control the scramble level according to the target object. The data structure of the scrambled image remains unchanged; therefore, no special viewer is necessary to view the scrambled image. We evaluated the scramble level controllability by using parameters and processing speed of this method in an embedded system.

II. Hierarchical Image-Scrambling Method

Our proposed hierarchical image-scrambling method has three special features: *restoration of the original image from only the scrambled image and its key*, *controlling of the scramble-level by using parameters for the length of random number*, and *opening the image with a general image viewer*.

A. Privacy-protecting surveillance system

Figure 1 shows an example application of protecting private information using the proposed method. The surveillance system using the proposed method is for an indoor environment. For example, we can recognize individuals from their face, and the PC monitor might display confidential information. Therefore, the system scrambled people's faces and the PC monitor. Normally, scrambled images are displayed on a monitoring console, and saved in storage. When there is a problem with the monitoring area, the system supervisor can authorize access to the scrambled area with scramble keys. The authorized individuals can only access the necessary information; therefore, this method prevents private information from being opened to the public.

B. Parameters for scramble level control

Our proposed method can control the scramble level by using three parameters for each color component. Table I lists these parameters. We set these parameters to each color component (e.g., Y/Cb/Cr for JPEG image, R/G/B for bitmap image). The JPEG format divides an image into 8×8 blocks for processing, then parameters *Start_coef* and *End_coef* can be set in JPEG format. The values *Start_coef* and *End_coef* mean the first and last coefficient to scramble in "zigzag" ordering.

The quality of a scrambled image is affected by parameters *Start_coef* and *End_coef*. A block's information is concentrated to low-frequency components by discrete cosine transform (DCT); therefore, scrambling low-frequency components takes more effect in concealing visual information than scrambling high-frequency components. The number of coefficients (determined by Parameter *Start_coef* – *End_coef*) affects the security of a scrambled image. One coefficient has only a few bits of information due to quantization; therefore, scrambling less coefficients could enable the image to be restored through brute-force attacks.

TABLE I

Parameters for Scramble Level Control

Parameter	Value	Description
Start_coef	0 – 63	Start coefficient in 8x8 block (For JPEG image only)
End_coef	0 – 63	End coefficient in 8x8 block (For JPEG image only)
Bit_length	0 – 8	Bit length of cipher pseudorandom number

TABLE II

Information for One Scramble (Quadrilateral)

	Information	Description
	s_ID	Scramble ID of this area
Repeat as many as components	Start_coef	Start coefficient in 8x8 block (For JPEG image only)
	End_coef	End coefficient in 8x8 block (For JPEG image only)
	Bit_length	Bit length of cipher pseudorandom number
	Start_x	X-coordinate of start pixel
	Start_y	Y-coordinate of start pixel
	End_x	X-coordinate of end pixel
	End_y	Y-coordinate of end pixel

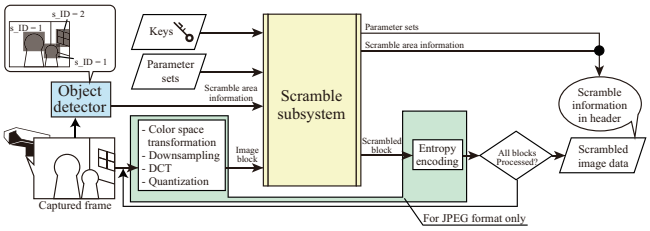
C. Header format for scramble information

Information used for scrambling is stored in the image header. Therefore, the scrambled image can be restored only with the correct key. Table II lists this information for one quadrilateral scramble area. The scramble ID *s_ID* and parameters *Start_coef*, *End_coef*, and *Bit_length* of each color component (e.g. Y/Cb/Cr for JPEG image), and area coordinates *Start_x*, *Start_y*, *End_x*, and *End_y* of each scramble are required to properly restore an image.

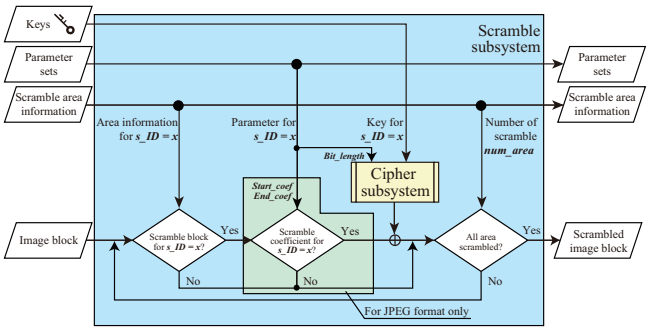
D. Processing flow

To produce a scrambled image that can be opened without a specific image viewer, the proposed method retains the image format structure. In this section, the processing flow of this method is discussed.

1) Scramble subsystem and scramble flow: Figure 2(a) shows the processing flow of scrambling a captured frame. First, an object (e.g., face, human body, window, or car license plate) is detected using an object detector to determine the area of the object that is to be scrambled. In this figure,



(a) Overview of processing flow



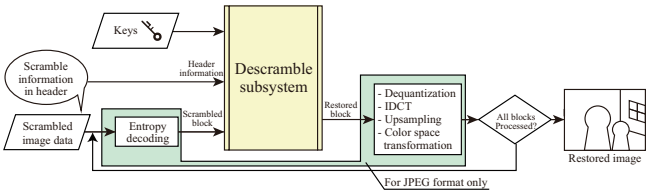
(b) Details of scramble subsystem

Fig. 2. Processing flow of scrambling image

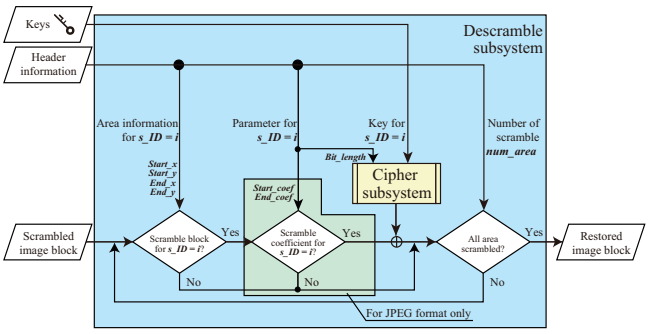
Detector 1 for face detection ($s\_ID = 1$) and Detector 2 for window detection ($s\_ID = 2$) are used. If both the camera and target object to be scrambled are fixed, the system supervisor can manually input the coordinates. The system supervisor must specify as many parameter sets (mentioned in Section II-B) and cipher keys as there are object detectors. Then, the image pixels (for bitmap image format) or coefficients (for JPEG format) are processed in the scramble subsystem. Since the conversion of a raw image to JPEG format is irreversible due to DCT and quantization, scrambling occurs between quantization and entropy coding processes for the JPEG format. Parameter sets and scramble area information are stored in the image header. This enables the restoration only with the correct cipher key and scrambled image data.

The processing flow in the scramble subsystem is shown in Figure 2(b). The image pixels (for bitmap image format) or coefficients (for JPEG format) are XORed with the cipher pseudorandom number stream if they are in the scramble area. Parameter Bit_length determines the bit length of the pseudorandom number ($0 - 8$ bit) and controls the scramble level. In JPEG compression, each coefficient in 8×8 blocks is EXORed between quantization and entropy coding processes.

2) Descramble subsystem and restoration flow: Figure 3(a) shows the processing flow for restoring an original image. After a supervisor sets the cipher key, this method analyses the scramble information from the data header. From this information, the descramble subsystem (shown in Figure 3(b)) restores the original image by XOR operation between the scrambled (encrypted) pixel value and cipher pseudorandom number. If the supervisor sets the wrong cipher key or does not have a



(a) Overview of processing flow



(b) Details of descramble subsystem

Fig. 3. Processing flow of restoring image

cipher key, the scrambled image cannot be restored since the cipher pseudorandom numbers are uniformly distributed.

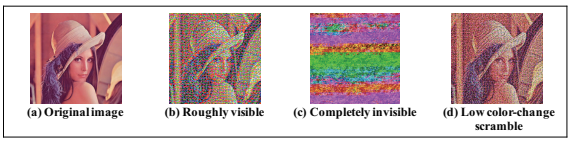
III. Experimental results

We discuss the correlation of the parameter sets and scramble-level controllability. We evaluated the processing speed of our method in a demo system. We use a modified libjpeg-turbo library [8] in this experiment.

A. Scramble level controllability

We scrambled the entire area of "Lena" image in various parameter sets (JPEG format, 512×512 pixels, $quality = 75$, file size of 38,710 bytes without scramble) to evaluate scramble-level controllability. Figure 4 shows that we can control the scramble level (the degree of intensity). With parameter set (b), the image is roughly visible, but there is no detailed information for distinguishing the individual. In parameter set (c), the image is completely invisible. JPEG handles luminance and chrominance as an image color space. Luminance represents brightness and chrominance represents color difference, usually in two components. Therefore, we can control color change by applying different parameter sets to both components (Figure 4-(d).

Figure 5 shows clippings of scrambled JPEG images. We applied the same parameter sets to the brightness and chrominance components. As can be seen from the figure, the particle size of the scramble becomes small when applying large End_coef parameter. It is because low-frequency coefficients (small value of coefficient parameter) in 8×8 block have information of the outline, and high-frequency coefficients (large value) have the detailed information.



Parameter sets {Start_coef, End_coef, Bit_length}			
for Luminance	{1, 4, 6}	{0, 4, 8}	{1, 9, 6}
for Chrominance	{1, 4, 6}	{0, 4, 5}	{1, 9, 3}
PSNR [dB]	9.70	7.21	11.59
File size [Byte]	60,471	72,459	81,510

Fig. 4. Scrambled images and parameter sets

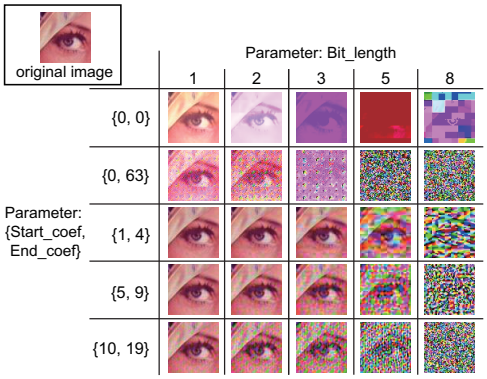


Fig. 5. Scramble-level transition (JPEG image)

To quantitatively observe the scramble level transition, we used the peek signal-to-noise ratio (PSNR) as an index. The PSNR transition of scrambled images (Figure 5) derived from parameters Bit_length, $Start\_coef$, and End_coef is shown in Figure 6. We applied the same parameter sets to each component. The PSNR decreased when parameter Bit_length increased. The PSNR also decreased in proportion to the number of coefficients ($End\_coef - Start\_coef + 1$).

B. Demo system and its processing speed

We developed a demo system to confirm that our method is applicable to embedded systems such as those equipped with surveillance cameras. The system uses a single board computer (BeagleBoard-xM with ARM Cortex-A8 processor 1 GHz and running the Ubuntu 12.04 operating system) and a USB web camera (Figure 7-(a)). We confirm that this method can scramble several areas hierarchically (Figure 7-(b)).

The average processing time of the proposed method in the demo system when scrambling an image of a face is shown in Figure 8. In this experiment, we used OpenCV's object detection function based on a Haar-like feature [9], and the image size was 640 × 480 pixels. It took about 500 ms on average to process one frame. Object detection took up high

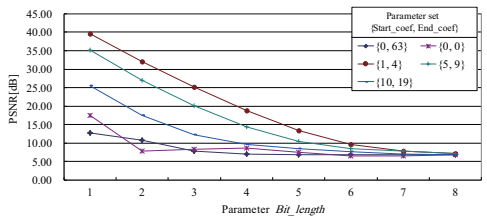


Fig. 6. PSNR transition of scrambled "Lena" image (JPEG format)

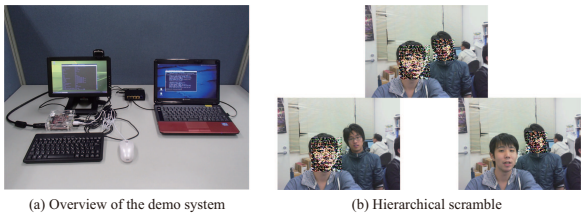


Fig. 7. Demo system and scrambled image

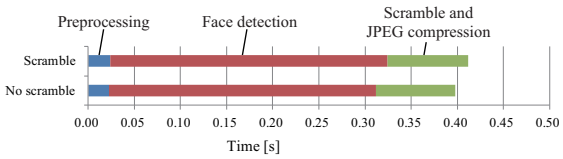


Fig. 8. Average processing time in demo system

percentage of the processing time. This is because a software-based object detector incurs a high-load on an embedded CPU; therefore using hardware-based object detector may be a solution to shorten processing time. When we use a hardware object detector (ASIC or FPGA), the increase in processing time compared to non-scrambling is about 1.4%. Therefore, our method can be applied to embedded systems such as those equipped with surveillance cameras.

IV. Conclusion

We proposed a light-weight and secure hierarchical image-scrambling method. It can be used to protect privacy by scrambling images, and the information of scramble area and parameter sets is included in the image. This allows restoration of the original image from only scrambled image and its key. The image format structure is retained; therefore, it can be opened with a general image viewer. Furthermore, our evaluation indicated that our method can flexibly control of scrambling with parameter sets regardless of contents in an image. There is little increase in processing time compared to non-scrambling when scrambling a face in an image; therefore, it can be applied to embedded systems such as those equipped with surveillance cameras.

References

[1] I. T. Almalkawi, et. al., "Wireless multimedia sensor networks: current trends and future directions," *Sensors*, vol. 10, no. 7, pp. 6662–6717, 2010.

[2] T. Honda, et. al., "Development of low-power camera sensor node using infrared array sensor and cmos image sensor," in *NCSP 2013*, 2013, pp. 508–511.

[3] L. Amoore, et. al., "A Report on the Surveillance Society," the Surveillance Studies Network, Tech. Rep., 2006.

[4] C. F. Shu, et. al., "Ibm smart surveillance system (s3): a open and extensible framework for event based surveillance," in *IEEE Conference on Advanced Video and Signal Based Surveillance, 2005. AVSS 2005.*, 2005, pp. 318 – 323.

[5] A. Senior, et. al., "Blinkering surveillance: Enabling video privacy through computer vision," *IBM Research Report*, vol. 22886, 2003.

[6] K. Yabuta, et. al., "Privacy Protection by Masking Moving Objects for Security Cameras," *IEICE Trans. Fundamentals*, vol. 92, pp. 919–927, 2009.

[7] H. Fujii, et. al., "Partial-scrambling of information," *NTT review*, vol. 11, no. 1, pp. 116–123, 1999.

[8] [Online]. Available: http://libjpeg-turbo.virtualgl.org

[9] [Online]. Available: http://opencv.org