A VIDEO SCRAMBLING SCHEME APPLICABLE TO LOCAL REGION WITHOUT DATA EXPANSION

Makoto Takayama, Kiyoshi Tanaka*, Akio Yoneyama** and Yasuyuki Nakajima***

* Faculty of Engineering, Shinshu University, Japan
** KDDI R&D Laboratories Inc, Japan

ABSTRACT

Recently, several scrambling techniques have been proposed for video digitally archived. These methods realize efficient processing by partial encryption on MPEG compressed data while keeping compatibility to MPEG format. However, they have common drawbacks expanding the entire code. Also, they have not reported on local shuffling in a frame. In this work, we propose a new video scrambling scheme on MPEG compressed domain, which shuffles a part of DC and AC coefficients by DCT in a frame. Our scheme is applicable to local region without data expansion from the original MPEG file.

1. INTRODUCTION

Scrambling is a well-known technique that makes image data meaningless visually. However, conventional methods for analog data cannot be used for video digitally archived, because most moving pictures are recently compressed in the form of MPEG that is the international standard for video coding. Therefore, a new approach directly applicable to MPEG compressed domain has been greatly needed for various applications such as video delivery on the internet. So far, several approaches have been proposed for MPEG video. Spanos and Maples [1] proposed to restrict MPEG encryption only to I-pictures that are the basis for predicting P and B-pictures. However, this scheme sometimes reveals visual information because of the presence of I-blocks in P and B-pictures. Meyer and Gadegast developed an encryption system called "SECMPEG" (see [2]), which provides four stages of encryption (only header encryption; encryption of header, DC and first AC coefficients of I-frames; all I-blocks encryption; full encryption). However, this is not compatible to MPEG because of special file format. Zeng and Lei [3] proposed a method to encrypt sign bits of motion vectors (MVs) with permutation of MVs. Although this scheme achieves fast shuffling, the encryption causes data expansion. In the similar period, Shi et al. [4] proposed to encrypt not only sign bits of MVs but also sign bits of DCT coefficients. Wen et al. [5] proposed a method to encrypt VLC (Variable Length Codes) table of MV. However, this scheme also causes small data expansion. Wang et al. [6] proposed a DCT-based transparent scrambling method, which is designed to show unauthorized users sufficiently degraded video rather than unrecognizable one. Since they perform scrambling by modifying quantized DCT cofficients before VLC in MPEG encoder, we cannot apply this scheme to video already achieved. Scrambling methods on MPEG compressed domain [3]-[5] are excellent in terms of efficient processing by partial encryption of MPEG compressed data while keeping compatibility to MPEG format. However, they have common drawbacks expanding the entire code. Also, they have not reported on local shuffling in a frame.

From this point of view, in this work, we propose a new video scrambling scheme on MPEG [7] compressed domain, which shuffles a part of DC and AC coefficients by DCT in a frame. We consider the following conditions to design our scheme: (i) applicable to local region in a frame, (ii) no data expansion from the original MPEG file, (iii) compatible to general MPEG decoder (iv) 100% reversible to the original video. Through computer simulation, we verify the basic performance of the proposed scheme.

2. MPEG CODING ARCHITECTURE

In the following discussion, we describe MPEG-2 method as one of popular MPEG coding schemes. In MPEG-1 or MPEG-2, input frames are classified into three kinds of picture, i.e. I-picture, P-picture and B-picture. I-picture is independently encoded within itself by removing intra-frame redundancy. The entire frame is first divided into small blocks consisting of 8×8 pixels. Then each block is transformed to 8×8 frequency components by DCT, and the coefficients after quantization are encoded. DC components are separately encoded by predictive coding for deferential signals between neighbor blocks. 63 pieces of AC component (excluded DC coefficient) are encoded by using 2-dimensional Huffman codes in a certain scanning order (e.g. Zigzag scan). On the other hand, in P-picture and B-picture encoding, interframe redundancy is removed by one-way or both-way motion compensation between frames, respectively. The motion compensation is conducted for every macro block (MB) consisting of 16×16 pixels. Then a motion vector (MV) is searched for a minimum error between the current MB and the same size block in a specified region on a reference frame. If the minimum error E_{min} is greater than a predefined threshold T, intra-coding is applied to MB. Otherwise, the differ-

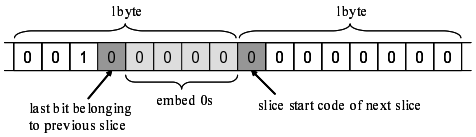


Fig. 1. An example of byte-alignment

ential signals are divided into four blocks consisting of 8×8 pixels, and their frequency components by DCT are quantized and encoded by using 2-dimensional Huffman codes similar to the encoding of AC components in I-picture. The only difference in P-picture and B-picture coding is that the DC and AC components are coded together. In MPEG encoding, byte-alignment is conducted at sequence header code, GOP start code, picture start code, and slice start code. **Fig.1** shows an example of byte-alignment at slice start code. In this example, 4 bits are embedded between the last bit belonging to the previous slice and the start code of next slice.

3. PROPOSED SCRAMBLING SCHEME

In the proposed scheme, we first specify the region R to be shuffled in a frame, and then determine blocks to be processed included in this region. The size of R is limited to the multiple of 8 in both horizontal and vertical direction since each block size is 8×8. Our scrambling scheme consists of (i) intra-slice shuffling of DC components, (ii) inter-block shuffling of AC components, and (iii) intra-block shuffling of AC components. (i) and (ii) can be applied only to I-picture but (iii) to all pictures. This scheme focuses on MPEG-1 and MPEG-2, but is applicable to MPEG-4 by some modification.

3.1. Intra-slice Shuffling of DC Components

Here we show the way of shuffling of DC components by using **Fig.2**. See a slice surrounded with a bold rectangle. In this slice, we have nine MBs in total. From the left-hand side, we have two MBs out of R (denoted as MB_{out}), five completely included in R (MB_{in}), and one that half of MV is included in R (MB_{half}), and one out of R as shown in **Fig.2**(a). If all (four) Y-blocks in MB are completely included in R (MB_{in} case), we shuffle all the DC components of Y together with the components of C_b and C_r. Otherwise, we do not shuffle any components because of the scanning order of four DC components in MB. As shown in **Fig.3**, four components are scanned in order of (left, top), (right, top), (left, bottom), and (right, bottom) in MB, and the differential signal between neighbor components is encoded. Therefore, any change in case of MB_{half} (partial inclusion) causes some effect to the region outside R. Inside the area *R*, all the differential signals of DC components of MB_{in} are shuffled by using a secret key K_{DC} as shown in **Fig.2**(b). We can avoid coded data increase because we shuffle DC components within the same slice. The original DC components in MB_{in} can be reconstructed by the inverse operation using K_{DC}.

3.2. Inter-block Shuffling of AC Components

Next we show the way of inter-block shuffling of AC components by using **Fig.4**. The region shown in **Fig.4** is corresponding to R in **Fig.2**. AC component blocks (denoted as B_{AC}) consisting of 63 elements (64 elements in P and B-picture) are independent each other. Therefore, shuffling all the B_{AC} inside the region R are never affected each other. In the proposed scheme, as shown in **Fig.4**, all B_{AC} of Y components together with C_b and C_r components (only in MB_{in}) inside R are fully shuffled by using a secret key $K_{AC}^{(inter)}$. The original order of B_{AC} can be reconstructed by the inverse operation using $K_{AC}^{(inter)}$.

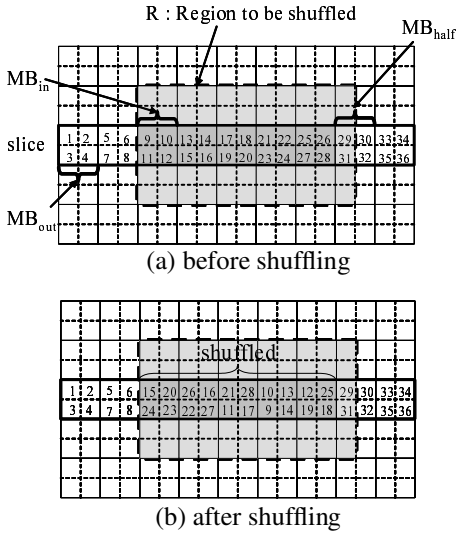


Fig. 2. Intra-slice shuffling of DC components (Y)

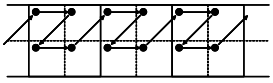


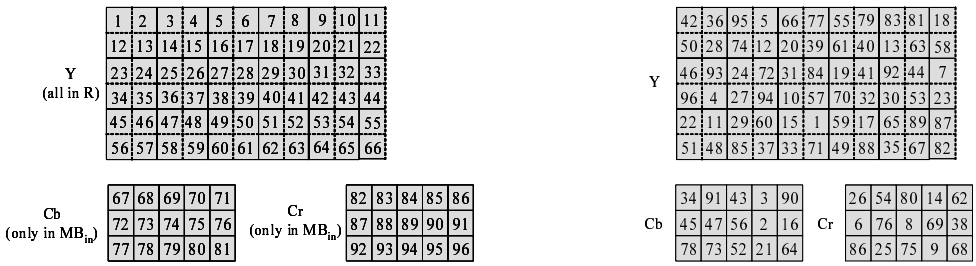
Fig. 3. Scanning order of four DC components in a MB

3.3. Intra-block Shuffling of AC Components

All elements inside B_{AC} are scanned with a certain order (e.g. Zigzag scan), and decomposed into pairs of the length of zero-run (denoted as *Run*) and the level of non-zero component (*Level*). **Fig.5**(a) shows an example of AC components inside B_{AC}, where we have six pairs of (*Run*, *Level*). A codeword in the predefined table of 2-dimensional Huffman codes is allocated to each pair of (*Run*, *Level*). Therefore, shuffling the order of (*Run*, *Level*) does not change the amount of codes allocated to each pair of (*Run*, *Level*). In the proposed scheme, the order of (*Run*, *Level*) for AC components in every B_{AC} is shuffled by using a secret $K_{AC}^{(intra)}$ as shown **Fig.5**(b). The original order of (*Run*, *Level*) can be completely recovered by the inverse operation using $K_{AC}^{(intra)}$.

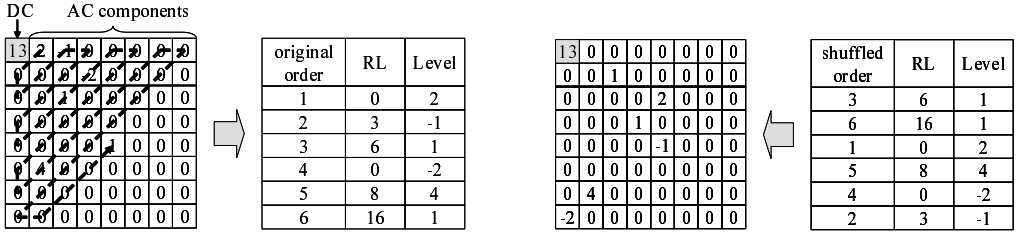
3.4. Compatibility and Secrecy

The modified MPEG file by the proposed scrambling scheme is completely compatible to general MPEG decoder because no MPEG syntax is modified nor additional header information and/or special code is used. Therefore, we can playback the scrambled MPEG file as it is. On the secrecy of the pro-



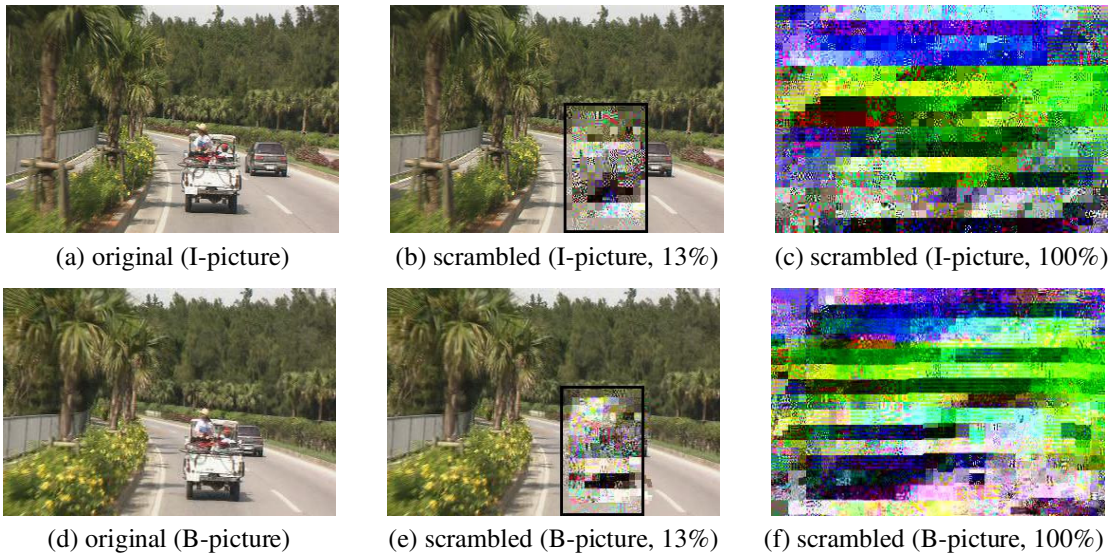
(a) before scrambling (b) after scrambling

Fig. 4. Inter-block scrambling of AC components



(a) original AC components (b) shuffled AC components

Fig. 5. Intra-block shuffling of AC components



(a) original (I-picture) (b) scrambled (I-picture, 13%) (c) scrambled (I-picture, 100%)

(d) original (B-picture) (e) scrambled (B-picture, 13%) (f) scrambled (B-picture, 100%)

Fig. 6. Examples of scrambled frame (C_R=1.5[Mbps])

posed scheme, it depends on the total permutation numbers in intra-slice shuffling of DC components, and inter-block and intra-block shuffling of AC components. Since the number becomes a huge one, it would be difficult to completely reconstruct the original MPEG file without knowing key information. All the keys, K_{DC}, $K_{AC}^{(inter)}$ and $K_{AC}^{(intra)}$, should be shared in both sender and recipient sides in advance, or should be authenticated before content delivery.

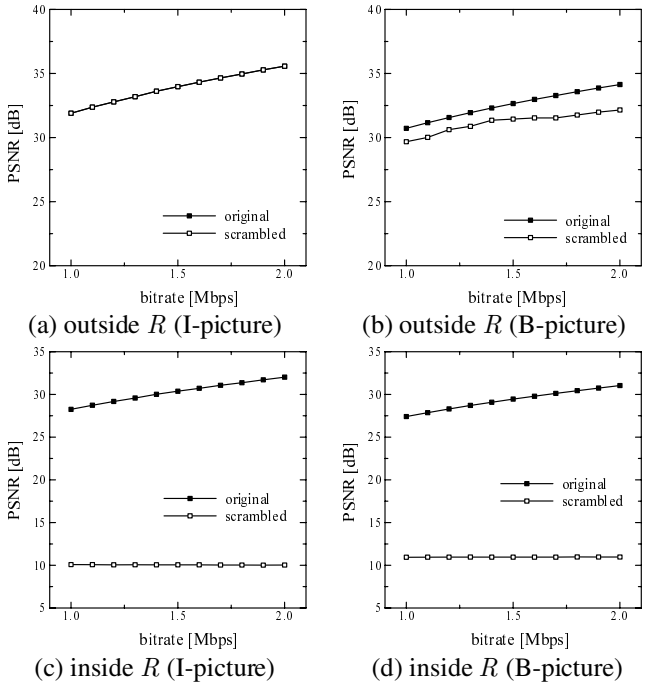
4. EXPERIMENTAL RESULTS AND DISCUSSION

To verify the basic performance of the proposed method, we conducted computer simulation by using some encoded videos: (i) "Driving" (352 $\times$ 240 pixels, 300 frames), (ii) "Flower Garden" (352 $\times$ 240 pixels, 150 frames), and (iii) "Mobile Calendar" (352 $\times$ 288 pixels, 300 frames). We implemented our scheme in MPEG-1 [7], where coding rate is set in the range C_R=1.0 $\sim$ 2.0[Mbps]. In this paper we only show the results for "Driving". First we show several examples of scrambling in **Fig.6**, where a local region (13% in the frame, specified by a bold rectangle) and the entire frame are shuffled by our scheme at C_R=1.5[Mbps]. Both I and B-pictures are shown with their original frames. We can see that it is difficult to visually understand the content of the shuffled regions. Comparing I-picture with B-picture in case of local shuffling, it can be observed that scrambling is also appeared outside the scrambled region in B-picture, which is mainly caused by inter-frame operation using motion compensation.

Table 1. Increase/decrease of code in each slice [Bytes]

	I-1	I-2	I-3	I-4	I-5	I-6	I-7	I-8	I-9
slice-1	0	0	0	0	0	0	0	0	0
slice-2	0	0	0	0	0	0	0	0	0
slice-3	0	0	0	0	0	0	0	0	0
slice-4	0	0	0	0	0	0	0	0	0
slice-5	0	0	0	0	0	0	0	0	0
slice-6	0	0	0	0	0	0	0	0	0
slice-7	-83	-56	-36	-20	-53	-80	-65	-43	-64
slice-8	42	32	8	14	3	-69	-27	-131	-52
slice-9	-40	-23	-48	-11	-72	-42	-59	16	-85
slice-10	-100	-73	-29	-28	-67	-67	-28	-45	-64
slice-11	-79	-72	-89	-62	-83	-43	-56	-96	-47
slice-12	8	16	25	5	83	72	77	102	88
slice-13	-39	-23	-10	-25	3	1	-27	-11	-21
slice-14	58	10	23	38	27	50	76	72	48
slice-15	232	187	158	88	159	180	110	137	197
total	-1	-2	2	-1	0	2	1	1	0

Next, we show PSNR of scrambled images for original ones as we change coding rate C_R in **Fig.7**, where two kinds of PSNR are measured outside/inside scrambled region R. For the region outside R, PSNR measured is exactly the same to the original one in case of I-Picture, while we can see slight deterioration in case of B-picture. However, for the region inside R, very low PSNR is maintained for all the coding rate. Next, we examine the statistics on the effect to MPEG code by the proposed scheme. There is a possibility to change the amount of coding bit only in the process of inter-block shuffling of AC components because of byte-alignment in each slice. To obtain statistics, we conducted 5×10^3 times of simulation by changing secret keys. **Fig.8** shows the total increase/decrease[bytes] from the original code. From this result, we can see that the effect is quite small and negligible in our scheme. We further observe the change in **Table 1**, which shows the increase/decrease of code in slice level. We can see various changes in slices included in R. However, they are canceled each other in a frame, and totally keep same MPEG file size. Finally, we show the increase of processing time (actual CPU time, no programming optimization) of our scheme as we increase the region to be scrambled in a frame in **Fig.9**. From this figure, our scheme linearly increases CPU time as we increase the region with the maximum of 55% for the entire frame.



(a) outside R (I-picture) (b) outside R (B-picture)

(c) inside R (I-picture) (d) inside R (B-picture)

Fig. 7. PSNR of scrambled frame

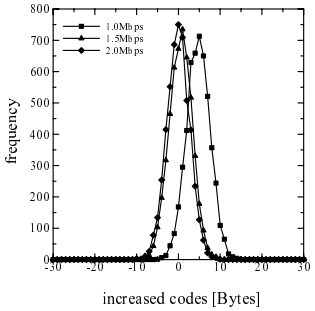


Fig.8. Total increase / decrease of MPEG code (local shuffling)

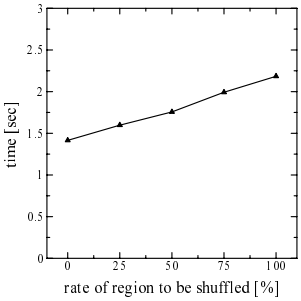


Fig.9. Actual processing time (Pentium IV, 3.0[GHz])

5. CONCLUSIONS

We have presented a video scrambling scheme on MPEG compressed domain, which is applicable to local region in a frame. Our scheme consists of (i) intra-slice shuffling of DC components, (ii) inter-block shuffling of AC components, and (iii) intra-block shuffling of AC components. Through computer simulation, we have verified that we can efficiently shuffle a part of frame without data expansion.

As future works, we should compare our scheme with other existing schemes, and further enhance the effect of scrambling. Also, we should suppress the effect of scrambling outside the scrambled region especially in P and B-pictures.

6. REFERENCES

[1] G. Spanos, T. Maples, "Performance study of a selective encryption scheme for the security of network real-time video". *Proc. 4th Int'l Conf. on Computer Communication and Networks*, 1995.

[2] L. Qiao, K. Nahrstedt, "Comparison of MPEG encryption algorithms", *Int'l Jour. on Computers and Graphics*, vol.22, pp.437-444, 1998.

[3] W. Zeng, S. Lei, "Efficient frequency domain video scrambling for contest access control", *Proc. 7th ACM Int'l Multimedia Conference*, pp.285-293, 1999.

[4] C. Shi and B. Bhargava, "A fast MPEG video encryption algorithm", *Proc. 6th ACM Int'l Multimedia Conference*, pp.81-88, 1998.

[5] J. Wen, M. Severa, W. Zeng, M. Luttrell, W. Jin, "A format compliant configurable encryption framework for access control of video", *IEEE Trans. on Circuits and Systems for Video Technology*, Vol.12, pp.545-557, 2002

[6] C. Wong, H. Yu, M. Zheng, "A DCT-based MPEG-2 Transparent Scrambling Algorithm", *IEEE Trans. Consumer Electronics*, vol.49, no.4, pp.1208-1213, 2003.

[7] ISO/IEC MPEG-1 Video encoder version 1.5g.