COMPRESSED-DOMAIN SCRAMBLER/DESCRAMBLER FOR DIGITAL VIDEO

Mohan S. Kankanhalli and Teo Tian Guan
School of Computing, National University of Singapore
Kent Ridge, Singapore 119260
E-mail: mohan@comp.nus.edu.sg

ABSTRACT

Multimedia data security is very important for multimedia commerce on the Internet and real-time video multicast. However, traditional encryption algorithms for data secrecy such as DES may not be suitable for multimedia applications because they are unable to meet the real-time constraints required by the multimedia applications. For multimedia applications, lightweight encryption algorithms are attractive.

This paper examines the joint encryption and compression framework in which video data are scrambled in the frequency domain by employing selective bit scrambling, block shuffling and block rotation of the transformed coefficients and motion vectors. In addition a lightweight MPEG video encryption algorithm is proposed whose primary motivation is to save the encryption computation by taking the advantage of combining MPEG compression and data encryption, and at the same time avoids adverse effects on the video compression rate.

Keywords: Multimedia data security, MPEG encryption, MPEG codec, digital video broadcasting, conditional access system.

1. INTRODUCTION

The proportion of free access channels among analog TV transmissions is decreasing continuously, and at the same time as the number of digital TV channels increases, a significant number of them will be pay-TV services. As MPEG-2 becomes the standard for digital video broadcasting, encryption of the MPEG-2 video data is important to provide assurance to the provider of pay-TV services that their investment will be worthwhile.

Data security will be extremely important for enabling multimedia commerce on the Internet such as pay-per-view digital video broadcast. In a typical pay-per-view broadcast system, digital video is transmitted through the coaxial cable or carried over-the-air, as described in Figure 1.1. The commercial digital video broadcaster needs to get the assurance that unauthorized users will not be able to gain access to its services. Hence, digital video signals have to be broadcast in an encrypted form, and can only be decrypted by means of a set-top box for an authorized user. The set-top box requires the necessary hardware, software, and interfaces to select, receive, and decrypt the programs. However, it must be kept at a minimum cost for the user.

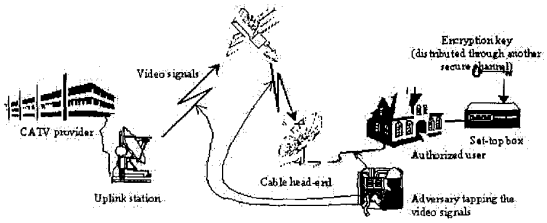

Figure 1.1 Tapping of digital video over a broadcast network.

Traditional cryptographic algorithms for data security are often not fast enough to process the vast amount of continuous data, such as video, to meet the real-time constraints. Most algorithms either added huge amount of overheads to both the encoding and decoding processes, or have adverse effects on the video compression efficiency. This causes unnecessary decoding delays, requiring high cost set-top boxes. Hence, lightweight encryption algorithms that do not add large amount of overheads, and yet provide reasonable security are preferred.

The rest of the paper is organized as follows: Section 2 presents some background information pertaining to the work, describing the MPEG-2 standard and Huffman encoding scheme. In addition, some related works on video data encryption and the encryption methods that this study was based on are described. The details of the proposed scrambling scheme are presented in Section 3. Experimental results and findings are shown in Section 4. Finally Section 5 makes some concluding remarks and discusses the future work that can be done.

Contributed Paper

Original manuscript received December 27, 2001

Revised manuscript received April 5, 2002

2. BACKGROUND

2.1 Overview of MPEG

MPEG is the acronym for the Moving Picture Experts Group, which was formed by the ISO (International Standards Organization) in 1990. The first outcome of its work was the ISO/IEC 11172, widely known as MPEG-1. MPEG-2 was then published in November 1994, and is the source-coding standard used by the European Digital Video Broadcasting (DVB) committee [10], and by the American ATSC (Advanced Television Systems Committee) for future digital TV broadcasting.

To help with error handling, random search, editing, and synchronization, MPEG defines a hierarchy of layers within a video sequence. A pictorial representation of the MPEG layers is given below.

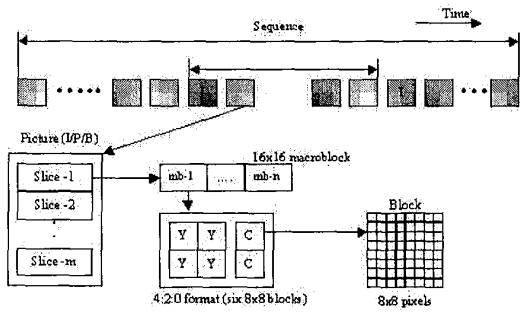

Figure 2.1 Hierarchy of the MPEG Video layers.

During encoding, successive groups of digital video information are reordered in order to benefit from the statistical correlation within successive groups of information. This process is followed by *discrete cosine transformation* (DCT). Next, the DCT coefficients are quantized, after which redundancy reduction by means of a run-length amplitude/variable length coder is applied.

To exploit temporal redundancy due to the strong correlation between successive parts of the digital video signal, 3 types of pictures are defined and arranged as shown:

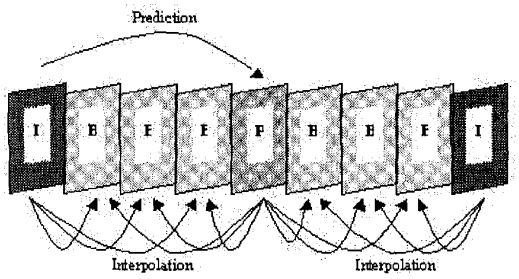

Figure 2.2 Types of pictures defined in MPEG-2.

I (intra) pictures are coded without reference to other pictures.

P (predicted) pictures are coded with reference to a past picture (I or P picture).

B (bi-directional) pictures require both a past and a future reference for prediction.

In addition to temporary redundancy reduction, MPEG exploits the spatial redundancy presence in the still images and prediction error signals to achieve high compression. The technique to perform intraframe compression with the DCT consists of three stages: computations of transform coefficients; quantization of the transform coefficient; and conversion of the quantized coefficients into {run, amplitude} pairs after reorganizing them in a zigzag or alternate scanning order (see Figure 2.3).

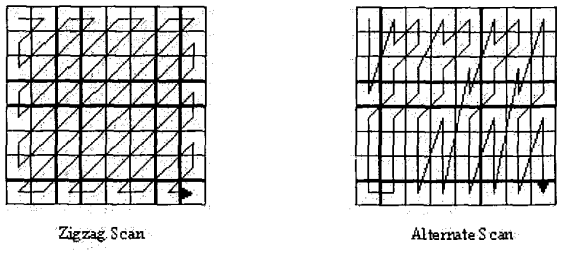

Figure 2.3 Methods for scanning DCT coefficients.

Scanning of the coefficients in a zigzag pattern results in a sequence of numbers with high probability of long runs of consecutive zero coefficients. This sequence is then represented as a run-length (representing the number of consecutive zeroes) and amplitude (coefficient value following a run of zeroes). These values are then looked up in a fixed table of variable length codes, where the most probable occurrence is given a relatively short code, and the least probable occurrence is given a relatively long code. It uses the Huffman coding algorithm.

If the individual code words of a Huffman Code are concatenated to form a stream of bits, then correct decoding by a receiver requires that every combination of concatenated code words be uniquely decipherable. A sufficient condition for this is that the code satisfies the so-called *prefix rule*, which states that no code word may be the prefix of any other code word. It is this prefix rule that renders the scrambling of the VLC (Variable Length Coding) tables difficult.

2.2 Related Work

The most direct approach to encrypt data is to employ known encryption algorithm such as Data Encryption Standard (DES). However, a DES implementation is not fast enough to process the vast amount of multimedia data due to the complicated computations involved. Furthermore, in these systems, the encryption/decryption processes are done in between MPEG encoding/decoding. This adds latency to real time video delivery.

Tang [15] proposed shuffling the DCT coefficients within an 8×8 block for JPEG/MPEG based transmission system. Tang's methods use a random permutation list to replace the zigzag order to map the DCT coefficients to a 1×64 vector. Since mapping according to the zigzag order and mapping according to a random permutation list have the same computational complexity, the encryption and decryption add very little overhead to the video compression and decompression processes. Unfortunately, this method decreases the compression rate significantly for some video sequences as noted in [15]. The reason being the statistical property of the DCT coefficients is changed.

Another scheme proposed by Shi and Bhargava [12] uses a permutation of the Huffman code word list as a secret key. During MEPG encoding (decoding), their encoder (decoder) uses the secret key instead of the standard Huffman code word list. Video frames decoded using the standard Huffman code word become incomprehensible. However, this method has a disadvantage in its difficulty in the generation of the encryption keys, resulting in having each generated key to be tested before using, as reported in [12].

Zeng and Lei [16] proposed a joint encryption and compression framework in which video data are scrambled in the frequency domain by employing selective bit scrambling, block shuffling and block rotation of the transformed coefficients and motion vectors. Their proposal performs encryption of the video data by scrambling the block coefficients and/or motion vectors.

A cryptographic key will be used to control the scrambling. The scrambled coefficients and motion vectors are then subjected to video compression before they are transmitted over networks or stored in a storage device. (In other words, the scrambling is performed prior to compression [quantization and Huffman, run-length, arithmetic, embedded, or other entropy coding.])

At the decoder, the compressed video bitstream will first be decompressed (entropy decoding and de-quantization). Authorized users will then use the same key to unscramble the decompressed coefficients before they are subjected to inverse transformation, and unscramble the motion vectors before motion compensation.

As mentioned in [16], a combination of the scrambling methods renders a completely incomprehensible image, and it is difficult for an adversary to recover the image through exhaustive searches. Nevertheless, these advantages are achieved at high computational cost. For example the implementation of block shuffling introduces unnecessary latency to the encoding/decoding process, as the whole slice of macroblocks have to be stored into memory, shuffle/unshuffle before the blocks of coefficients can be written to output file. Besides with a combination of all the suggested scrambling methods ([Random sign change + Block shuffle shuffling] on all pictures + Random sign change on the motion vector), the compressed file size increases by as much as 10%.

An encryption scheme that provides reasonable security with less computational overhead, and yet does not affect the compression rate adversely is required. Hence, this provides the motivation for the suggested scrambling method.

3. SCRAMBLING OF THE HUFFMAN CODE WORDS

As mentioned in the previous section, the proposed approach aims to reduce the overheads that are added to the encoding/decoding processes during encryption/decryption. In addition, it does not increase the compression ratio adversely. These twin aims are achieved by applying the techniques proposed in [16] on the Huffman tables that are used for Variable Length Coding (VLC). The resulting proposal is simple to implement; yet, as we will show, it provides reasonable security.

The Huffman encoding scheme can be considered as a *substitution algorithm*, where during encoding, it transforms (with a function **H**) the original video data **M** into variable length bit stream **B** with a key **k**. At the decoder end, a function **D** is used to obtain the original data from the bitstream with the use of the same key.

$$\mathbf{M = D\,(H\,(M, k), k)} \qquad \text{---------- } ①$$

Of course, in a normal MPEG2 video, the functions H, D and the key k are already fixed. Hence we can simplify equation ① to **M = D (H (M))**

The function **D** that MPEG decoder uses is to map each H(M) to an entry in the table(s) to decipher the variable length bits. This function is best illustrated with the example of MPEG Table B-12 (Table 3.1) showing the entries in the VLC tables used in decoding and encoding:

VLC code word	DCT-DC luminance size	Entry in encoder table		Entry in decoder table		
		{value, length}	Table index	{size, length}	Table index	
9*1	11	{0x01ff, 9}	11	{11, 9}	15	
8*1 0	10	{0x01fe, 9}	10	{10, 9}	14	
7*1 0	9	{0x00fe, 8}	9	{9, 8}	12 – 13	Table II
6*1 0	8	{0x007e, 7}	8	{8, 7}	8 – 11	
5*1 0	7	{0x003e, 6}	7	{7, 6}	0 – 7	
4*1 0	6	{0x001e, 5}	6	{6, 5}	30	
1110	5	{0x000e, 4}	5	{5, 4}	28 – 29	
110	4	{0x0006, 3}	4	{4, 3}	24 – 27	
101	3	{0x0005, 3}	3	{3, 3}	20 – 23	
01	2	{0x0001, 2}	2	{2, 2}	8 – 15	Table I
00	1	{0x0000, 2}	1	{1, 2}	0 – 7	
100	0	{0x0004, 3}	0	{0, 3}	16 – 19	

Table 3.1 Entries used for the encoding and decoding from MPEG Table B-12.

The proposed encryption algorithm encodes the video with new functions **H'**, and decoding requires the corresponding **D'**. Keys are used to generate these H' and D' functions. **D'** actually is the reshuffling of the indexes of the code words in the tables.

3.1 The Proposed Algorithm

The proposed scrambling algorithm consists of two methods:

1. Shuffling of the code words within each of the Huffman tables.
2. Random flipping of the last bit of the code word.

3.1.1 Shuffling of the Code Words for VLC

The code words used for VLC are shuffled within each Huffman table during the initialization of the encoding process. To minimize the impact on the compression rate, the code words for each of the tables are grouped into different subsets according to their bit length of code words. Different shuffling tables are generated for each Huffman table, increasing the difficulty of guessing the correct sequence. Similarly, when the correct values are obtained after re-shuffling during the decoder's initialization process, the entire video can be decoded with any further changes to the original program design.

In addition, this scrambling method has a nice property that any shuffling permutation will not violate the prefix rule of Huffman coding. Therefore, there is no need for extra checks for the validity of the shuffling table to ensure that the decoder is able to decipher the concatenated code words.

Algorithm to shuffle the code words:

Let $\mathbf{C_i}$ be the code word used to represent th original information $\mathbf{O_i}$ in Table $\mathbf{T_j}$, and **k** be th scrambling key. Let **E** be the shuffling operatio (the same shuffling table is used for encryption an decryption).

During encoding/encrypting, **E** $(\mathbf{C_i}, \mathbf{k})$ gives a new code word to represent each O_i in Table T_j. For proper decipherment during th decoding/decrypting process, a function **D'** i required to map O_i to the correct entry in th corresponding decoder tables for T_j.

D' (E (C_i, k)) is used to find the correc index in the decoding table.

Table 3.2 Algorithm for shuffling the Huffman code words.

3.1.2 Random Bit Flipping of the Code Words for VLC

To increase the level of security, the last bit representing the code word is randomly flipped. However, to ensure the prefix rule is not validated, flipping of the last bit may require changing the bits of those affected code words. Hence, the following rule is devised:

Rule for random bit flipping:

For each code word C_i in the table,

Flip its last bit randomly.

If C_i last bit is flipped (C_i becomes C_i'), then

For each subsequent code word C_j in the table,

If C_i' becomes the prefix of C_j, then

Replace the affected prefix of C_j with Ci

End if C_i' becomes the prefix of C_j

End for each subsequent code word

End if last bit is flipped

End for each code word

Table 3.3 Rule for random flipping of the last bit of the Huffman code word.

To illustrate, suppose the last bits for all the code words in Table B-12 are flipped, the new Huffman table is as shown in Table 3.4.

Original code words in MPEG Table B-12	New code words	Size
1 1111 1111	1 1000 0000	11
1 1111 1110	1 1000 0001	10
1111 1110	1100 0001	9
111 1110	110 0001	8
11 1110	11 0001	7
1 1110	1 1001	6
1110	1101	5
110	111	4
101	100	3
01	00	2
00	01	1
100	101	0

Table 3.4 Code words after implemented the rules for bit flipping.

As the example in Table 3.4 has illustrated, the new Huffman tree retains the same shape as the original. This implies the video (entropy coded with this new tree) retains the statistical property (the size of the video will not be increased nor will the compression rate be decreased). Unauthorized users who use the original code words to decode will not be able to restore the picture.

3.2 Analysis of the Proposed Scrambling Methods

The proposed algorithm adds little overhead to the encoding/decoding process. Compared to the extensive DCT operations overheads, the processing amount required for the scrambling and unscrambling of the Huffman code words is very low.

3.2.1 Security Strengths/Weakness

Because a single bit error in the transmission will scramble all future transmissions (an effect called "error propagation"); an adversary will have to guess all the correct shuffling positions in order to recover the complete picture. However, experiments conducted have shown that not all tables with their code words scrambled produce visually scrambled images. This is expected, as only those tables (namely from Table B-12 to Table B-15) that are involved in the entropy encoding of the DCT coefficients will affect the image. Let the cardinality of each set of Huffman code words scrambled be n_i. The number of permutations for the shuffling of the Huffman code words is:

$\prod_{i=1}^{k}((n_i)!)$, where k is the total number of subsets of Huffman cod words shuffled across all tables.

For the second scrambling method, the number of permutations can be calculated as follows:

$\prod_{i=1}^{h} 2^{m_i}$, where h is the number of Huffma tables whose the last bits of the cod words are randomly flipped. m_i is total number of possible code words in tabl i whose last bit can be flipped. (Eac of these code words can be represente in two forms: its origina representation or the representation with its last bit flipped)

A quantitative breakdown showing the total number of permutation is as follows:

MPEG Table	Number of possible scrambling
B-10[1]	$3! \times 4! \times 9!$
B-12	$7! \times 5! \times 2^6$
B-13	$6! \times 6! \times 2^8$
B-14	$6! \times 13! \times 4! \times 4! \times 6!$
B-15	$20! \times 16!$
B-14 and B-15[2]	$16! \times 16! \times 4! \times 12! \times 10! \times 10!$
Total	$\mathbf{1.68 \times 10^{123}}$

Table 3.5 Total number of tries required ensuring a perfect image is recovered.

As not all code words in the tables will be used for the encoding of an image, some subsets of the code words scrambled within a table will produce little unintelligible images to a viewer. Further experiments carried out show that certain sets of code words do scramble the image consistently.

MPEG Table	Number of possible scrambling
B-10	$3!$
B-12	$7! \times 2^6$
B-13	$6! \times 2^8$
B-14	$6!$
B-15	$16!$
Total	$\mathbf{5.37 \times 10^{27}}$

Table 3.6 Estimated number of scrambling methods.

The above is a conservative estimate of the minimum number of combinations that the proposed scrambling methods can offer to distort the video image. The effect on the scrambled image is dependent on the particular distribution of the coefficients in the spatial domain of the video which really depends on its content.

Since the code words are actually bit strings representation of the original codes and the first scrambling method changes the length of the bit string representation, the compression ratio will be increased slightly. However, this actually makes the algorithm more resistant to plaintext attacks. Scrambling the Huffman code words, without changing the length of the code words is very vulnerable to plaintext attacks. As by obtaining some of the original video images (the *plaintext*) and the encrypted video (the *cipher text*), an adversary can determine all the substituted code words easily by performing an alignment of the encrypted bit stream with the original. Shuffling the code words within the Huffman tables, coupled with the random flipping of the last bits of the code words, provides good *confusion* and will discourage random guessing.

Scrambling the Huffman code words once for the entire video implies that once the adversary obtains the unscrambled Huffman tables by launching a successful plaintext attack, the entire video can be deciphered without further efforts. Hence, to overcome this weakness, and to further increase the security of the proposed algorithm, the Huffman code words can be reshuffled after certain number of frames. This number can be treated as a unicity distance that is the number of frames required to uncover all the scrambled Huffman code words.

3.2.2 Other Considerations

Some encryption algorithms encrypt every bit of the video bit stream, including the long run of zeros for resynchronization and the MPEG header bits. With these bits encrypted, the video will not be readable to common MPEG tools. Since the proposed scrambling methods do not change the MPEG video structure, an encrypted file is still readable by MPEG tools. This is an important feature as a nonpaying watcher can be enticed to subscribe to this service provided by the conditional access TV provider, an idea that is first presented in [16].

4. EXPERIMENTAL RESULTS AND FINDINGS

Experiments are conducted primarily to:

1. Compare the performance of different scrambling schemes.

2. Evaluate the effects of different scrambling schemes has on the video.

Two performance metrics are used to gauge the effects of the encryption has on the video files:

1. Size of scrambled video file. The study of this effect is important as entropy coding is one of the compression schemes employed by MPEG, and the proposed algorithm is working on the Huffman code words used for entropy coding. Ideally, there should be no effect on the compression ratio.

[1] Scrambling Table B-10 (table for motion vector code) will distort the motion of the video.

[2] Some code words are common between Table B-14 and Table B-15. Hence, these code words have to be scrambled together.

2. Time taken to decipher the scrambled video file into its raw PPM files format. The purpose is to provide an idea on the overheads caused by the proposed algorithm and the existing encryption schemes. Again, it is desirable that the scrambling and descrambling processes have no impact on the time taken for this operation.

All results are obtained from a DELL computer running on Microsoft Windows NT 4.0, with dual PIII-450 processors and 256 MB RAM.

4.1 Effects of Different Scrambling Techniques

As it is impossible to include all the images frames here, only the effects of I-frame scrambled using different techniques are shown (Figure 4.1a and 4.1b). Table 4.1 provides a description of the effects.

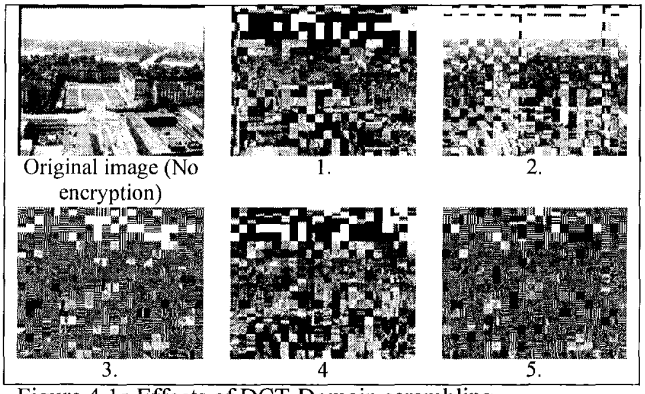

Figure 4.1a Effects of DCT-Domain scrambling.

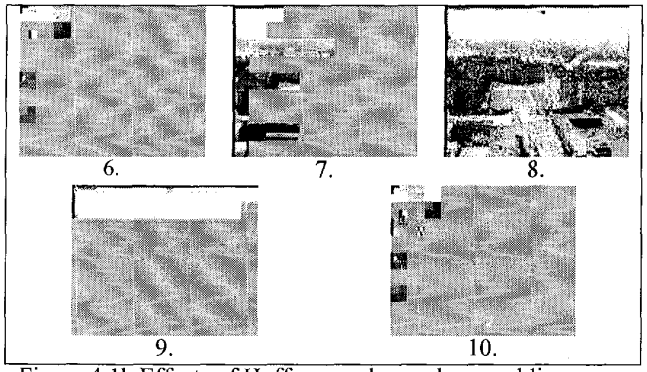

Figure 4.1b Effects of Huffman code words scrambling.

Another scrambling technique is selective scrambling of different types of frames. In this experiment, the effect of scrambling employing random sign change (block coefficients), block shuffling along slices, block coefficients rotation and random sign change (motion vector) on the I-frame alone is compared with that on all frames-types. A group of images selected from a GOP of each of the scrambled videos is displayed in Figure 4.2.

Scrambling Methods Used	Visual effects on the video image
1. Random sign change (block coefficients)	Though outlines of objects are visible, generally provides unintelligible frames.
2. Block shuffling along slices	Different objects in the video are still distinguishable.
3. Block coefficients rotation	Produces unintelligible images, but motion is visible if the motion vectors are not scrambled.
4. Random sign change (block coefficients) + Block shuffling along slices	Produces unintelligible images.
5. Random sign change (block coefficients) + Block shuffling along slices + Block coefficients rotation + Random sign change (motion vector)	Produces unintelligible images.
6. Shuffling + Flipping of last bit of code words in Table B-12	A large portion of the image is indistinguishable.
7. Shuffling + Flipping of last bit of code words in Table B-13	Some portions of the original image still remain visible.
8. Shuffling of code words in Table B-14	A large portion of the original image is still visible.
9. Shuffling of code words in Table B-15	Produces an unintelligible image.
10. Shuffling + Flipping of last bits of code words in Table B-12 and B-13 + Shuffling of code words in Table B-10, B-14 and B-15	Produces an unintelligible image.
11. Shuffling of code words in Table B-1, 3, 4.	Not significant impact on the image (results not displayed here).
12. Shuffling of code words in Table B-10 alone. OR Random sign change (motion vector)	Picture content is highly visible except that motion of the picture is distorted (results not displayed here).

Table 4.1 Description of scrambling effects.

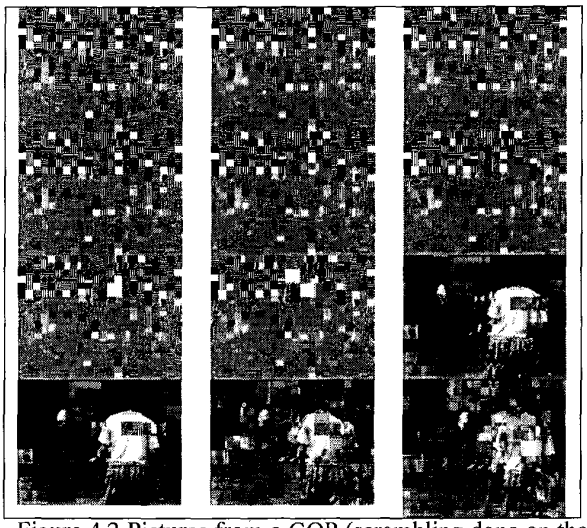

Figure 4.2 Pictures from a GOP (scrambling done on the I-frames).

Some of the pictures, as shown in Figure 4.2, are still distinguishable if only the I-frames are scrambled. However, this can entice a non-subscriber to pay for the services provided by the commercial digital video broadcaster. Furthermore, the overhead cost of scrambling all frame types is found to be too expensive compared to scrambling of only the I-frame. Hence, if the requirement on the security aspects of the video is not that critical, then it is recommended to scramble only the I-frames using the method proposed in [16].

4.2 Scrambling of Different Sub-Sets of Code Words

As not all sub-sets of Huffman code words will produce incomprehensible images when they are subjected to scrambling, experiments are conducted to test which sub-set produces consistent scrambling effects. This study is important, as an adversary can forgo the efforts of descrambling a sub-set, in exchange for a movie that is still viewable, though not that enjoyable (as some portions of the picture are still encrypted).

The effect of scrambling different sub-sets of the Huffman code cords on one frame from the "Carphone" video sequence is shown in Figure 4.3. The cardinality of each sub-set is indicated within the brackets. As the effects of scrambling each sub-set of code words are highly dependent on the spatial frequency distribution the DCT coefficients in the video pictures, the effects shown in Figure 4.3 are not meant to be representative. However, they provide us with a gauge on how effective the scrambling each of the sub-sets.

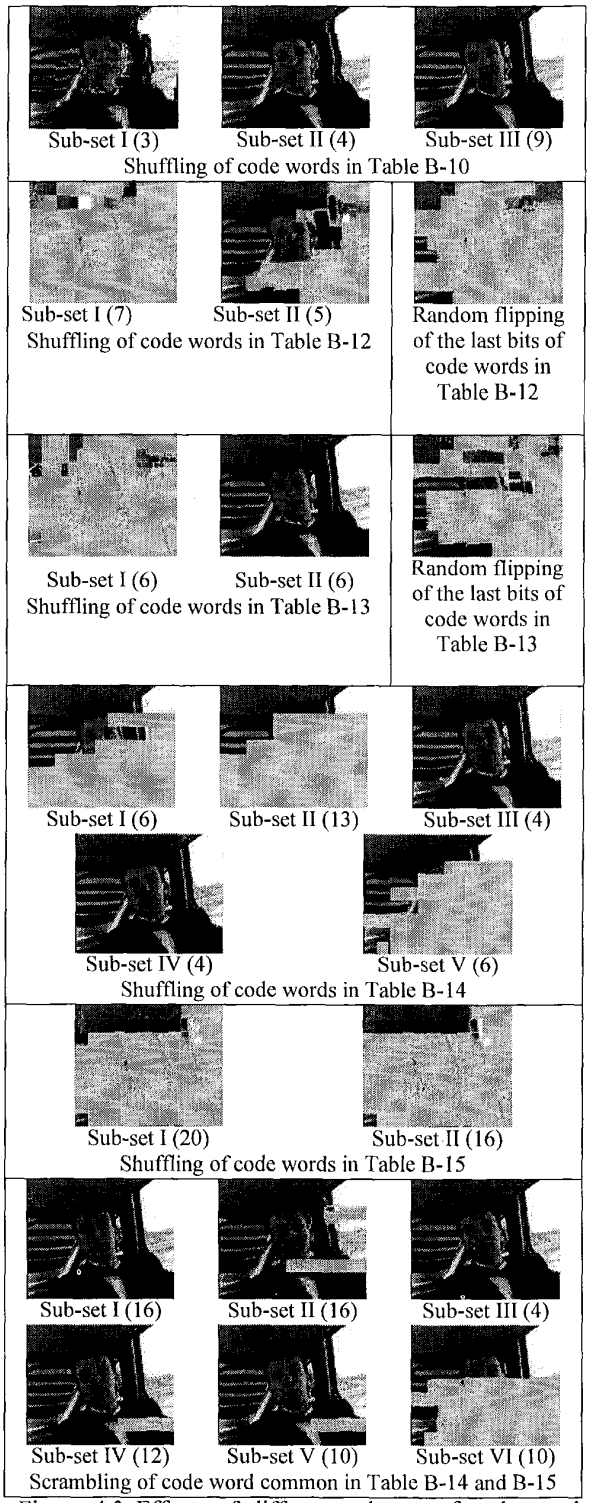

Figure 4.3 Effects of different sub-sets of code words being scrambled.

In general, scrambling of those code words in Table B-12 and Table B-13 renders the video picture

incomprehensible. Sub-set I from Table B-14 and sub-set II from Table B-15 produce consistent scrambling effects on the video image too. The shuffling of code words that are common to both Table B-14 and Table B-15 yield the least effect on the picture. Scrambling of the code words for motion vectors mainly distorted images that are "predicted" by motion interpolation, and hence have no effect on encryption of I frames.

4.3 Performance of Different Scrambling Techniques

Keeping all environment settings consistent for each of the test video sequence, experiments were conducted to observe the impact of the proposed scrambling technique as compared to the existing scrambling methods. The time taken to unscramble the video is calculated and the size of the scrambled files are measured and compared in the Table 4.2.

Each video sequence are at 704 x 480 with 450 frames		Original video	Existing Scrambling Technique[3]		Scrambling on Huffman code words[4]
			I frames	All frame types	
"Susi" sequence	File Size	11,250,590	11,250,362 (-0.002%)	11,250,154 (-0.003%)	11,250,321 (-0.002%)
	Time	728	748 (+2.75%)	939 (+29.0%)	729 (+0.14%)
"Cactus &Comb" sequence	File Size	11,249,649	11,249,935 (+0.003%)	11,249,579 (-0.001%)	11,249,851 (+0.002%)
	Time	733	755 (+3.0%)	947 (+29.2%)	734 (+0.14%)
"Mobile &Calend ar" sequence	File Size	11,225,931	11,232,990 (+0.06%)	11,231,947 (+0.05%)	11,229,146 (+0.03%)
	Time	733	751 (+2.46%)	948 (+29.3%)	734 (+0.14%)
"Flowerg arden" sequence	File Size	11,258,616	11,261,370 (+0.02%)	11,260,466 (+0.02%)	11,269,610 (+0.10%)
	Time	731	750 (+2.6%)	945 (+29.3%)	732 (+0.14%)

Table 4.2 Performance of different scrambling technique.

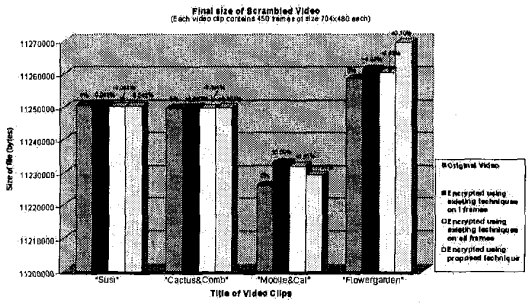

Figure 4.4 Graphical representation of the final size of the scrambled video.

[3] Random sign change (block coefficients) + Block shuffling along slices + Block coefficients rotation + Random sign change (motion vector)

[4] Shuffling + Flipping of last bits of code words in Table B-12 and 13 + Shuffling of code words in Table B-10, 14 and 15

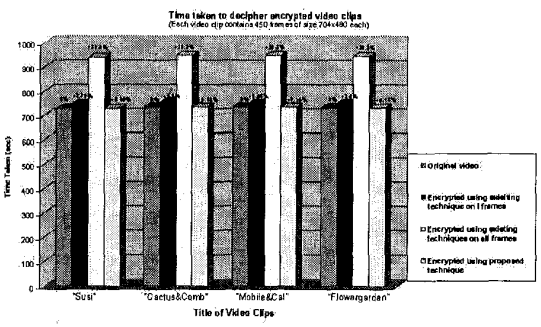

Figure 4.5 Graphical representation of the time taken to decipher the encrypted video

Generally the proposed scrambling technique will not increase the size of the compressed video, even though the Huffman code words used for entropy encoding are being shuffled, whereas the existing scrambling techniques frequently result in larger file sizes. Moreover, the proposed scrambling technique requires much less overhead when compared with the existing method that scrambled on all frame types (requires around 30% more time to unscramble). Even the time required unscrambling the file with only the I-frames encrypted seems larger than that of scrambling the Huffman code words. Therefore, it will be much cheaper to scramble video files using the proposed technique than employing the existing techniques. No quantitative results are shown here on the impact of the proposed technique on the picture quality, as only the Huffman code words used for entropy coding are scrambled, there will no loss in signal quality due to noise.

5. CONCLUSION

Macq and Quisquater[2] have pointed out that the value of multimedia information is much lower, while the bit rate is much higher. Hence, though our proposed scrambling method might be vulnerable to various cryptanalysis attacks the high costs of breaking our scheme (relative to the cost of buying the video) is sufficient to dissuade any adversaries. Contrarily, the high cost of encrypting the video data is of concern. Therefore, the low overhead cost of our algorithm compared to the scrambling of each of the block coefficients makes our scheme more attractive for a low-cost scrambling system that provide reasonable security.

In addition, the MPEG decoding process increases the cost of plaintext attack to most of the encryption algorithms. Coupled with the fact that a computer does not know if a guessed picture is comprehensible to a human being, human intervention is required for a cryptanalysis attack. These factors are sufficient to deter any adversaries from considering breaking of the encrypted video.

One area that can be explored is to develop an algorithm to generate all possible Huffman trees for each of the VLC tables that are used for entropy encoding (based on the exact number of bits that are used for each code word). Furthermore, efforts are required to ensure that all the Huffman trees generated can be used for proper encoding/decoding of the video. The work done here can be considered as a subset of this possible algorithm.

In addition, some of the video encoded in the experiments has shown improvement of the compression efficiency (in term of the compressed video sizes) when the code words in the VLC tables were permutated. Hence, further research can be done to improve the compression efficiency by dynamically changing the code words used for VLC, based on the distribution of the spatial frequencies during encoding and decoding.

Lastly, most of the research work done on the encryption of MPEG video works on the video layer, and testing is usually limited to local machines. It is best to develop a system layer scrambler/descrambler and carry out experiments to observe possible side effects that the encryption can cause when the MPEG video is broadcast over a network. The results of these experiments will be very useful, as MPEG-2 becomes the standard for digital television.

6. REFERENCES

[1] Ash Rofail, "Adopt a UI-driven architecture", http://www.devx.com/upload/free/features/entdev/1999/06jun99/fe0699/fe0699.asp, June 1999.

[2] B. Macq and J. Quisquater, "Cryptology for digital TV broadcasting," In Proc. Of the IEEE, vol. 83(6), pp. 944-957, 1995.

[3] Barry G. Haskell, Atul Puri, Arun N. Netravali, "Digital Video: An Introduction to MPEG-2", 1997, Chapman & Hall.

[4] Charles P. Pfleeger, "Security in Computing", 1989, Prentice-Hall.

[5] Didier Le Gall, "MPEG: A Video Compression Standard for Multimedia Applications," April 1991/Vol.34, No.4/ Communications of the ACM.

[6] H. Benoit, "Digital Television, MPEG-1, MPEG-2 and Principles of the DVB System", 1997, Arnold.

[7] J. Meyer and F. Gadegast, "Security Mechanisms for Multimedia Data with the Example MPEG-1 Video," http://www.gadegast.de/work.html, 1995.

[8] J. Watkinson, "MPEG-2", 1999, Focal Press.

[9] Jennifer Seberry, Josef Pieprzyk, "Cryptography: An Introduction to Computer Security", 1989, Prentice-Hall.

[10] Ronald de Bruin, Jan Smits, "Digital Video Broadcasting Technology, Standards, and Regulations", 1999, Artech House, Inc.

[11] Shi, C. and B. Bhargava, "A Fast MPEG Video Encryption Algorithm". In Proceedings of the 6th ACM International Multimedia Conference, Bristol, UK pp. 81-88.

[12] Shi, C. and B. Bhargava, "Light-weight MPEG Video Encryption Algorithm". In Proceedings of the International Multimedia Conference on Multimedia,(Multimedia98, Shaping The Future) January 23-25, 1998, pages 55-61, New Delhi, India. IETE, Tata Mcgraw-Hill Publishing Company.

[13] Shi, C., S. Y. Wang, and B. Bhargava: 1999, "Fast MPEG Video Encryption Algorithms", Purdue University, USA.

[14] Shi, C., S. Y. Wang, and B. Bhargava: 1999, "MPEG Video Encryption in Real-time Using Secret Key Cryptography". In Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA' 99), Las Vegas, Nevada, USA

[15] Tang. L.: 1996, "Methods for Encrypting and Decrypting MPEG Video Data Efficiently". In Proceedings of the ACM Multimedia96, pages 219-229, Boston, MA., November 1996.

[16] W. Zeng and S. Lei, "Efficient Frequency Domain Digital Video Scrambling for Content Access Control", Proceedings of the conference on ACM multimedia '99, 1999.

BIOGRAPHIES

Mohan S Kankanhalli got his B.Tech. in EE from Indian Institute of Technology, Kharagpur, M.S. and Ph.D. in Computer & Systems Engineering from Rensselaer Polytechnic Institute, NY. He is currently a faculty member with the School of Computing, NUS. His research interests are in Multimedia Information Systems and Information Security.

Teo Tian Guan received his B.S. degree in Computer Science from the National University of Singapore in 2000. He has since joined Adroit Innovations Limited as a Software Engineer specializing in applications for the Internet. Currently he is pursuing a Masters degree with the School of Computing, NUS as a part-time student.