Methods for Encrypting and Decrypting MPEG Video Data Efficiently

Lei Tang*

Oracle Corporation
500 Oracle Parkway
Redwood Shores, CA 94065
ltang@us.oracle.com

Abstract

Multimedia data security is very important for multimedia commerce on the Internet such as video-on-demand and real-time video multicast. However, traditional cryptographic algorithms for data secrecy such as DES are not fast enough to process the vast amount of data generated by the multimedia applications to meet the real-time constraints required by the multimedia applications. How to incorporate cryptographic technology with digital image processing technology to provide multimedia security does not seem to be considered in the previous literatures. The main contribution of this paper is the idea of incorporating cryptographic techniques (random algorithms) with digital image processing techniques (image compression algorithms) to achieve compression (decompression) and encryption (decryption) in one step. One of our methods is almost as efficient as the existing video encoding and decoding process while providing considerable level of security without affecting the quality of images when decrypted. Our methods are also adjustable to provide different levels of security for different requirements of the multimedia applications. Our methods are based on the widely used JPEG and MPEG standards. We also conduct a series of experimental studies to test and evaluate our algorithms.

Keywords: Multimedia security; multimedia encryption; compression; MPEG codec; multimedia commerce.

*The author is on leave of absence from the Ph.D program of GSIA, Carnegie Mellon University. This work was performed when the author was a Ph.D candidate in Carnegie Mellon University. The views and conclusions contained in this document are solely those of the author and should not be interpreted as representing the official policies, either expressed or implied, of Carnegie Mellon University, or Oracle Corporation.

1 Introduction

Multimedia applications that operate on audio and video data will enable new applications of computers. They will affect many aspects of our life. The applications include entertainments(e.g., video-on-demand, digital video broadcast, and interactive video games), multimedia email, documents, desktop video conferencing, and many more examples.

The distinction between the multimedia applications and the ordinary computer applications is the vast amount of data generated by multimedia applications and the real-time constraints required by the multimedia applications. One central ingredient of multimedia computing and processing is real-time execution to enable delivering and presentation of continuous synchronized media. While most research on multimedia applications has focused on compression standard, real-time communication, storage representation, and etc., little work had been reported on techniques for encrypting and decrypting multimedia data in real time, either due to the complexity of the problem, or due to protection of commercial secrecy.

The distinction between the multimedia information and the ordinary electronic information (e.g., bank information, high classified documents, etc.) is that the multimedia information value is very much lower. An very expensive cryptoanalysis attack of the scrambled multimedia data may not be interesting to the adversaries [7].

Secrecy of multimedia data is very important for multimedia commerce on the Internet. For instance, it is desirable in a video-on-demand system (or a digital video multicasting system) that only those computer users who have paid for the service can access those videos or movies. In a video conference system, only those members of the conference can watch the video.

Real-time constraints are required for many multimedia applications. The vast amount of data of these mul-

timedia applications put great burden on the encoding process (compression), the decoding process (decompression), and communication. Encryption (during or after the encoding phase) and decryption (during or after the decoding phase) will aggravate this problem and will increase the latency. How to encrypt and decrypt the vast amount of data efficiently is an important issue for multimedia security.

JPEG and MPEG (including MPEG-1, MPEG-2, and MPEG-4) are two widely accepted image compression standards. JPEG is the still picture encoding and decoding standard. MPEG is the motion picture standard. They are implemented and widely used for multimedia applications.

There are several proposed and implemented ad hoc methods for encrypting and decrypting multimedia data [9, 8]. Unfortunately, they are either too inefficient to meet the real time requirement [9], or too naive to meet the security requirement [8, 2] (the details are given in section 3).

Digital image processing techniques are used to enhence the quality of images, and to reduce the data needed to represent an image without visually affecting the quality of images (compression). Cryptographic techniques are used to scramble images so that an adversary could not obtain the original image without knowing the secret key. They have opposite effects. It is also commonly believed that the very nature of MPEG encoding–the encoding of inherently spatially and temporally correlated video–makes the encryption difficult [7].

We address the multimedia data[1] security in a novel way. We try to achieve compression (decompression) and encryption (decryption) in one step with minimum overhead to the encoding (decoding) procedure. We incorporate some cryptographic techniques such as random permutation, and probabilistic encryption, with the compression and decompression algorithms. We also provide different level of secrecy for different multimedia applications. One of our encryption (decryption) algorithms adds minimum computational overhead to the (JPEG and MPEG) encoding and decoding procedure while achieving a considerable level of secrecy for the multimedia data being processed. We believe that our algorithms will have wide applications for multimedia commerce on the Internet. Finally, we implement our algorithms into the Berkeley MPEG encoder [5]. We conduct a series of experimental studies to test and evaluate our algorithms based on the Berkeley MPEG decoder [10].

Our presentation is organized as follows. Section two describes the JPEG standard, MPEG-1 standard and the relationship between them. Section three describes the related previous works. In section four we describe how to encrypt and decrypt JPEG and MPEG-1 video data, then we adjust our algorithms according to our experimental results. We also give the analysis of the theoretical computational complexity to break our algorithm in this section. Section five presents the conclusion and open questions. We describe our experimental results in the appendix.

[1] In this paper, we use multimedia data to refer to the digital images encoded by the MPEG-1 standard.

2 Background

In this section we will describe the JPEG standard and the MPEG-1 standard. We explain the JPEG standard because the MPEG-1 standard uses the same compress techniques used in the JPEG standard. We detail the processes related to our algorithms and omit those unrelated details.

2.1 Introduction to JPEG

JPEG stands for "Joint Photographic Expert Group". Its goal is to develop an international standard for color and greyscale image compression. We do not describe the lossless JPEG mode since it is seldomly used or implemented. We do not describe the extended JPEG modes such as the progressive mode and the hierarchical model either. We only describe the most widely used mode–the baseline sequential mode (a lossy mode). For a detailed description of JPEG standard, the reader should refer to [13, 11]. The procedure for the baseline compression is shown in Figure 1.

We outline the baseline compression procedure as follows.

1. Transform the image into a suitable color space. This is a no-op for greyscale images. But for color images, the RGB space is usually transformed into the luminance/chrominance color space (YIQ, YCbCr, YUV, etc.).

2. **Discrete Cosine Transform**
   Group the pixel values for each component into 8x8 blocks, then map each 8x8 block to another 8x8 block through a discrete cosine transform (DCT). The definitions of DCT and inverse DCT are shown as follows.

$$F(u,v) = \frac{1}{4}C(u)C(v)[\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)* \cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}]$$

$$f(x,y) = \frac{1}{4}[\sum_{u=0}^{7}\sum_{v=0}^{7} F(u,v)*$$

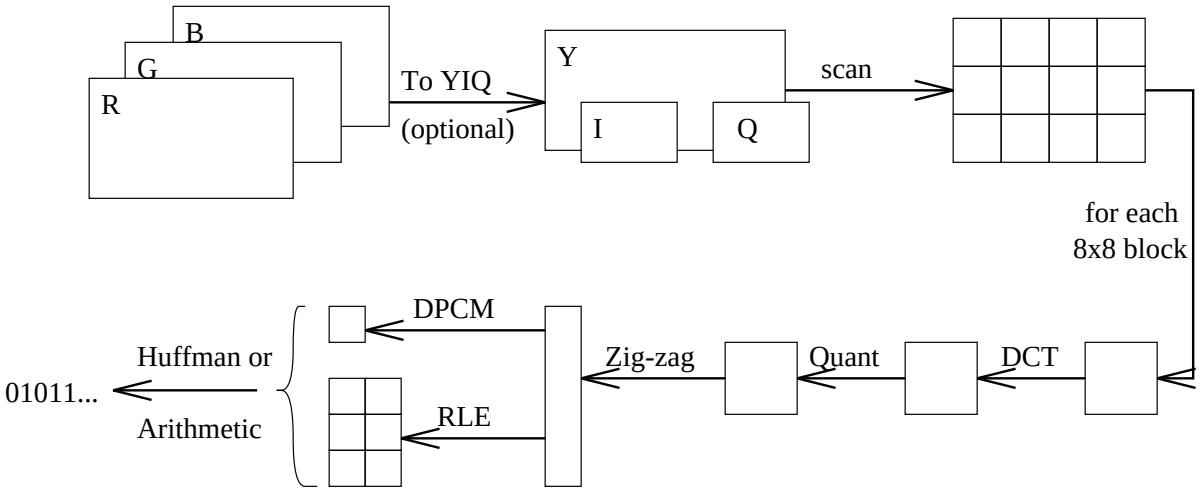

Figure 1: The JPEG compression procedure

$$\cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}]$$

where $C(u), C(v) = 1/\sqrt{2}$ for $u, v = 0$; $C(u), C(v) = 1$ otherwise.

The output of the DCT is the set of 64 basis-signal amplitudes or "DCT coefficients" whose values are uniquely determined by the particular 64-point input signals. The coefficient with zero frequency is called the "DC coefficient" and the remaining 63 coefficients are called "AC coefficients".

DCT has excellent energy compaction for highly correlated data. The signal energy of the 8x8 block is concentrated into a few coefficients (especially the DC coefficient) in the transformed block. The DC coefficient is a measure of the average value of the image sample. In our encryption algorithm, we will treat the DC coefficients differently from the AC coefficients.

3. **Quantization**
   After output from the DCT, each of the 64 DCT coefficients is uniformly quantized in conjunction with a 64-element Quantization Table, which must be specified by the application (or user) as an input to the encoder. The goal of this processing step is to discard information which is not visually significant. Quantization is a many-to-one mapping, and therefore is fundamentally lossy. The formula for quantization is shown as follows.

$$F^Q(u,v) = \lceil \frac{F(u,v)}{Q(u,v)} \rceil$$

   where $Q(u,v)$ is the (u,v)th quantizer in the quantization matrix Q. The quantization tables are usually put in image/scan headers.

4. **Zig-zag scan**
   The quantized DC coefficient is encoded as the difference from the DC term of the previous block in the encoding order. Since the DC coefficients frequently contain a significant fraction of the total image energy. Finally, all the quantized coefficients are ordered into a "zig-zag sequence". The zig-zag procedure is shown in Figure 2.

   The AC coefficients are coded by the Run Length Encode (RLE) method as (skip, value) pairs and $(0,0)$ as the end-of-block sentinel value.

5. **Entropy coding**
   This step achieves additional compression losslessly by encoding the quantized DCT coefficients more compactly based on their statistics. The JPEG proposal specifies two entropy coding methods–Huffman coding [6] and arithmetic coding [12]. Usually, the Huffman coding method is used since the arithmetic coding method is patented.

2.2 Introduction to MPEG-1

[2] MPEG stands for "Moving Picture Experts Group". There are three standards MPEG-1, MPEG-2, and MPEG-4 targeted for different applications. MPEG-1 has been implemented and widely used. It targets VHS quality video (320x240) with a CD quality audio at the speed of 1.5Mbits/second. It consists of three parts: video, audio, and system to control interleaving of streams. This section briefly describes the MPEG video coding model. More complete description are

[2] This section borrows heavily from [10]

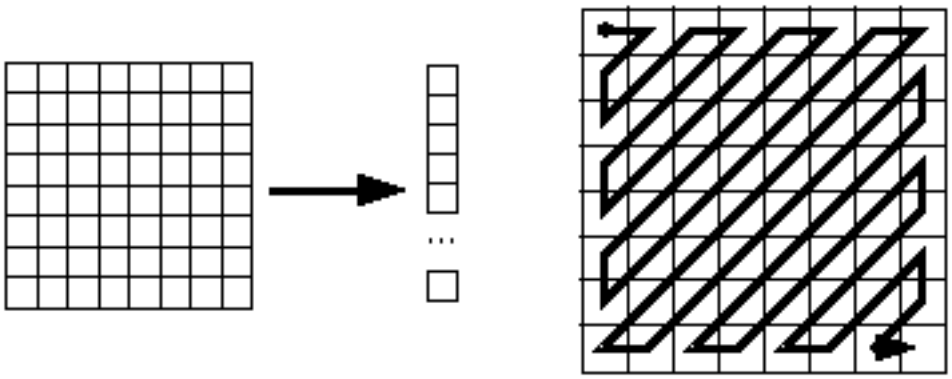

Figure 2: The zig-zag procedure

given in an introductory paper [4] and the ISO standard [1].

Video data can be represented as a set of images, $I_1, I_2, ..., I_N$, that are displayed sequentially. Each image is represented as a two dimensional array of *RGB triplets*, where an RGB triplet is a set of three values that give the red, green and blue levels of a pixel in the image.

MPEG video coding uses three techniques to compress video data. The first technique, called *transform coding*, is similar to JPEG image compression described in Section 2.

In MPEG transform coding, each RGB triplet in an image is transformed into a YCrCb triplet. The Y value represents the *luminance*(black and white) level and Cr/Cb values represent *chrominance* (color information). Since the human eye is less sensitive to chrominance than luminance, the Cr and Cb planes are *subsampled*.

Processing continues by dividing the images into *macroblocks*. Each macroblock corresponds to a 16 by 16 pixel area of the original image. A macroblock is composed of a set of 8 by 8 pixel *blocks*, four from the Y plane and one from each of the (subsampled) Cr and Cb planes. Each of these blocks is processed in the same manner as JPEG: the blocks are transformed using the DCT and the resulting coefficients quantized, run length encoded to remove zeros, and entropy coded.

The second technique MPEG uses to compress video, called *motion compensation*, exploits the fact that a frame I_x is likely to be similar to its predecessor I_{x-1}, and so can be nearly constructed from it. Three types of frames are considered in MPEG. Intraframe(I), Predicted frames(P) and Interpolated frames(B-for bidirectional prediction). When a large pool of reference frames is available, motion compensation can be used to construct more of the frame being encoded, reducing the number of bits required to encode the frame. A P frame is built an earlier frame. A B frame is built from both a preceding frame and a subsequent frame. An I frame is constructed using only transform coding. The I frames contains most of the signal energies (or information) of the pictures being encoded.

The final technique MPEG-1 uses to compress video data is *entropy coding*. After motion compensation and transform coding, a final pass is made over the data using Huffman coding.

3 Related works

Due to the protection of commercial secrecy, most of the commercial algorithms for scrambling digital images are not available to us.

In all of the known systems for scrambling digital images, the multimedia data are treated as ordinary bitstream and are encrypted (decrypted) using the traditional cryptographic algorithms such as DES [9]. But they still add a big portion of overhead to meet the 1.5Mbits/second requirement for the real-time video delivering (for MPEG).

To reduce the amount of data needed to be encrypted, a method called *selective encryption* was introduced in [8]. The basic idea of the selective encryption method is to encrypts only the I-frames of the MPEG video. It can reduce the amount of encryption and decryption. The security rational behind this proposal is that since I-frames carry the basic information, if they are unavailable to an eavesdropper, then the P- and B-frames become useless even though they are transmitted unencrypted. A series of experimental studies conducted by Agi and Gong [2] show that the intuitive rational behind this method is not correct. Moreover, this method still adds large computational overhead to the encoding and decoding processes.

4 Incorporation of Compression and Encryption

4.1 MPEG and JPEG image security

There are some fundamental differences between digital image information and text-based information. Digital image information has high information rate and low information value. The purpose of encrypting text information is to prevent an adversary without the secret key from obtaining the information. The encryption algorithm has zero-one semantics, the content of the information is either known completely, or is unknown completely. But there are two levels of security to digital images.

- The encrypted (or scrambled) image has poor image quality compared with the original image, but the content of the original image is visible to the viewer. We call this kind of scrambled image the *obscured image*.
- The contents of the scrambled image are not comprehensible to the viewer. We call this kind of scrambled image the *incomprehensible image*.

Due to the special property of digital images, it is unnecessary to treat digital images as bit streams and encrypt them bit by bit. New light-weight cryptographic algorithms should be invented. We give a list of desirable properties that an ideal digital image encryption algorithm should have.

1. Due to the high information rate of the digital image data, the computational overhead introduced by the encryption procedure and decryption procedure should be as low as possible.
2. The encryption procedure should not decrease the compression rate.
3. The encryption algorithms should be robust against common image processing techniques such as image enhencement, image restoration, etc.
4. The encryption and decryption of the digital images should not affect the quality of the original images.

We will evaluate our algorithms based on the above standards.

4.2 Overview of our approaches

It is commonly believed that the very nature of MPEG (also JPEG) encoding–the encoding of inherently spatially and temporally correlated video–makes the encryption difficult [7]. Contrary to this belief, our goal is to achieve compression (decompression) and encryption (decryption) in one step. The computational complexity of the encoder and the decoder must remain low enough to deal with the high information rate.

Our basic idea is to use a random permutation list to replace the zig-zag order to map the 8x8 block to a 1x64 vector. Mapping according to the zig-zag order and mapping according to the random permutation order has the same computational complexity. The secret key is the 1x64 permutation list. The difficulty of finding the secret key is introduced by the large number of inverse DCT operations in order to break our scheme by the ad hoc method (the details are given in Section 4.4).

There are some special properties unique to the JPEG and MPEG encoding (decoding) process to force us to make adjustments to the permutation list. We also conduct a series of experimental studies on the software-based MPEG encoder [5] and MPEG decoder [10]. We adjust our algorithm according to our experimental results.

After the discrete cosine transform and the quantization procedure, the coefficients of every 8x8 block has the following properties.

1. All coefficients are in the range of $[-255, 255]$, which can be denoted by an eight-digit binary string if the sign of the number is not considered.
2. The amplitude of the DC coefficient is much larger than that of every AC coefficient. A big portion of the AC coefficients are zeros (this is where the compression comes).

4.3 Our algorithm

Instead of mapping the 8x8 block to a 1x64 vector in the "zig-zag" order, we use a random permutation list to map the 8x8 block to a 1x64 vector. There are many known ways to produce a permutation list which has uniform distribution over all possible permutations and we will not go into details on this issue. Before we describe our algorithms, we conduct a series of experimental studies on the platform of the Berkeley MPEG encoder and decoder. All images from the experiments are shown in the appendix. We list our results as follows.

1. We map the DC coefficient to the first element in the 1x64 vector and randomly permute the rest 63 AC coefficients. The images are obscured but comprehensible under all some permutation lists we have used (Appendix, Figure 5).
2. We set the DC coefficient in every 8x8 block to be zero or a fixed value between 0 and 255, and

permute all the AC coefficients according to the zig-zag order, the skeletons of the images are obscured but comprehensible. The result is shown in Figure 6 of the appendix.

3. We map the DC coefficient to any position other than the first position in the 1x64 vector, then we randomly permute the AC coefficients, the images are completely incomprehensible. The result is shown in Figure 7 of the appendix.

4. We set the 63th (last) AC coefficient zero by adjusting the value of the corresponding element in the quantization table. We find that the degradation of image quality is negligible. The images are shown in Figure 8 of the appendix.

Recall that the amplitude of the DC coefficient is the largest one among all the coefficients. The position of the DC coefficient can be determined no matter which position we map it to in the 1x64 vector. If an adversary knows the position of the DC coefficient, he could switch the DC coefficient to the first position. Experiment one shows that the contents of images may be visible to an adversary after he switches the DC coefficient to the first position. This may unacceptable to some sensitive image information. Experiment two also shows that the knowledge of the position of the DC coefficient is important. If an adversary knows the position of the DC coefficient, he can set all the DC coefficients to zero and reduces the computational complexity to break the system. Experiment four shows that it may be possible to set the last AC coefficient to be zero while the degradation of image quality caused by this modification is negligible. Based on this observation, we set the last AC coefficient of every block to be zero.

Suppose that a DC coefficient is denoted by an eight-digit binary number $d_7 d_6 ... d_1 d_0$. We split the eight-digit binary number into two numbers $d_7 d_6 d_5 d_4$ and $d_3 d_2 d_1 d_0$, which both are in the range of $[0, 15]$. Then we set the value of the DC coefficient to be $d_3 d_2 d_1 d_0$ and the value of the last AC coefficient to be $d_7 d_6 d_5 d_4$ (remember that the last AC coefficient has value zero). We call this procedure the *splitting procedure*.

After the splitting procedure, the block is randomly permuted and the position of the DC coefficient is disguised. If the permutation list is know to a decoder, the decoder can permute all the coefficients back to their original position, then gets the value of the DC coefficient by concatenating the last element and the first element, and sets the value of the last AC coefficient to be zero.

So our algorithm consists of three steps.

1. Generate a permutation list with cardinality 64. This step can be done off-line.

2. Complete the splitting procedure after the 8x8 block is quantized.

3. Apply the random permutation list to the split block, and pass the result to the entropy coding procedure.

Our experimental results show that all scrambled images we have tested are incomprehensible (Figure 10 of the Appendix). The computational overhead to the encoding process is the splitting procedure, and the computational overhead to the decoding process is the concatenation procedure.

4.4 Analysis of our algorithm

Theoretically, it is difficult to recover the image without knowing the permutation list. After the splitting procedure, we divide the 64 elements into k different groups. All elements which has the same value belonging to same group. The cardinality of every group is denoted by $n_1, ..., n_k$ respectively. Here $\sum_{i=1}^{k} n_i = 64$. The number of different permutations for these coefficients is

$$\frac{(\sum_{i=1}^{k} n_i)!}{\prod_{i=1}^{k}(n_i!)}$$

which is

$$\frac{64!}{\prod_{i=1}^{k}(n_i!)}$$

We give a pessimistic estimation of the computational complexity to break the random permutation. We assume that there are 50 AC coefficients that are zeros. For the rest of thirteen AC coefficients, there are three distinctive elements, which has cardinality 6, 5, and 3 respectively. Moreover, we assume that all elements in the right lower triangle of the matrix are known to be zero after the quantization procedure. Then the number of different permutation is

$$\frac{(64-32)!}{(50-32)!6!5!3!}$$

which is $7.92 \cdot 10^{13}$. It is still computationally infeasible to try $7.92 \cdot 10^{13}$ inverse DCT operations and quantization operations in a reasonable amount of time. Please note that the complexity of obtaining the correct permutation list is increased by the number of the inverse DCT operations (the definition of IDCT is given in Section 2.1), not just the permutation list itself.

The bound of our estimation is not tight in the sense that we do not consider the special distribution properties of those AC coefficients, although the split of DC

coefficient will complicate the exact identification of the AC coefficients.

The permutation list is vulnerable to the known-plaintext attack. For instance, many MPEG videos starts with the MGM header–the roaring lion. If we encrypt the MGM header and other frames using just one permutation list, an adversary who knows the unscrambled MGM header images can obtain the permutation list by comparing the unscrambled MGM hearder images with the scrambled ones. Then the adversary can obtain all scrambled images using the same permutation list. To increase the security of our scheme, we propose two additional methods. One method increases the computational overhead of the encoding process and the decoding process slightly. One method increases the key length. We describe them in the next section.

4.5 Additional options

The DC coefficient of the 8x8 block includes most of the signal energy of this block. After the quantization process, the amplitudes of the signals are reduced and can be denoted by an 8-bits binary sequence[3]. If we apply a cryptographic function f to d, f must be a function mapping from $[1,...,255]$ to $[1,...,255]$. Otherwise, the coding length will be increased and the compression rate will decrease. We define a function f mapping $[1,...,255]$ to $[1,...,255]$ as follows.

Denote the DC coefficient of the ith 8x8 block of the encoding sequence by $x_{i1}...x_{i8}$ ($x_{ij} \in \{0,1\}$ for $j = 1,...,8$), where $0 < i \leq m$ (m is the number of the 8x8 blocks in an image). We group eight DC coefficients together to form a 64-bits binary sequence. The encryption(decryption) function f_i for encrypting(decrypting) the DC coefficient of the ith block is defined as follows.

$$f_i(x_{i1}...x_{i8}) = (DES_k(x_{11}...x_{18}.....x_{81}...x_{88}))_{8*i+1,...,8*i+8}$$

Here DES is the NIST data encryption standard and k is the 64-bits secret DES key and x_{ij} denotes the ij-th bit of the binary sequence x ($i,j = 1,...,8$). An alternative way to define the function f is given as follows

$$f_i(x_{i1}...x_{i8}) = (k \oplus (x_{11}...x_{18}.....x_{81}...x_{88}))_{8*i+1,...,8*i+8}$$

where k is a 64-bits secret key and $\oplus$ is the binary XOR function. There are many ways to group the eight DC coefficients. The eight DC coefficients can be grouped in sequential order, or in a random order. In our experiment, we encrypt every DC coefficient using the second function defined above before it is split and randomly permuted. All tested images are completely incomprehensible. The experimental result is shown in Figure 11 of the Appendix.

The whole image is as weak as a 8x8 block if we only use one random permutation list and apply it to every 8x8 block. Once the random permutation list is figured out, the whole image can be obtained. In order to increase the level of security of our algorithm. We introduce an additional technique. We can generate two (or more) permutation lists. For every 8x8 block, we flip a coin. If tail, the permutation list one is applied to this block. If head, the permutation list two is applied to the block. The binary coin flipping sequence, together with these two permutation lists, are the secret keys.

4.6 Discussion

The three cryptographic techniques can be applied independently or combinatorially to encrypt multimedia data according to different security requirements. If we combine the random permutation method with the probabilistic encryption method, we add minimum computational overhead (the splitting procedure or the concatenation procedure) to the encoding(compression) and decoding(decompression) process since the two permutation lists and the coin flipping sequence can be generated off-line.

Due to the well-defined structure of MPEG codec, and the special distribution property of the sixty-three AC coefficients, our methods might be vulnerable to various cryptoanalysis attacks. So they may not be ideal for those highly sensitive multimedia applications. However, they are enough for commercial applications in which the costs of breaking our schemes are much higher than that of buying the videos.

5 Experiments

We implement our algorithms on the platform of the Berkeley MPEG encoder [5] and test them using the MPEG decoder [10]. All the results are obtained from a SUN SPARCstation 20 running Solaris 2.4. We use two sequences of MPEG images to test and evaluate our algorithms: "table tennis" and "flower garden". It is impossible for us to include all the image frames here. Instead, we just show one frame from each video sequence to demonstrate our algorithms. All the images are shown in the appendix. We use the algorithm described in [3] to generate the permutation list although it does not have uniform distribution over all permutations.

[3]Here we assume the DC coefficient can be denoted by an 8-bits binary sequence. But this is not a requirement of our encryption method. This number can be adjusted in the real application. Our method by using DES is also vulnerable to a dictionary attack, we can chain the the encryption of different block using the CBC block

Experiment	Time (*sec*)	Size (*byte*)	Figure
original	37.985068	174387	5
1	37.434533	199934	6
2	37.634512	172152	7
3	37.949037	212744	8
4	37.264656	174387	9
5	37.920955	212602	10
6	37.969692	216030	11

Figure 3: Results for "flower garden"

Experiment	Time (*sec*)	Size (*byte*)	Figure
original	14.213096	31231	5
1	14.830879	37850	6
2	14.073405	30662	7
3	14.271419	44309	8
4	14.035959	31224	9
5	14.305303	44292	10
6	14.403090	45371	11

Figure 4: Results for "table tennis"

We conduct the following experiments on the MPEG encoder platform.

1. Map the DC coefficient to the first element in the 1x64 vector, then randomly permute the rest 63 AC coefficients.
2. Set the DC coefficient zero, then permute the AC coefficients according to the "zig-zag" order.
3. Randomly permute all coefficients and the DC coefficient is not in the first position of the vector.
4. Permute all coefficients according to the "zig-zag" order, then set the last AC coefficient zero.
5. Split the DC coefficient according to the splitting procedure, then permute all coefficients randomly.
6. Encrypting the DC coefficient by the second function defined in Section 4.5. Then apply the splitting procedure and the random permutation procedure.

We list the results obtained from the "flower garden" sequence of images in the first table and the "table tennis" sequence of images in the second table.

We find that the splitting procedure and the random permutation procedure add almost no computational overhead to the encoding process. Moreover, the encoding time under some permutation is less than the time used to encode the images using the standard MPEG encoding procedure.

We also observe that the compression rate is slighly decreased under some permutations in the "flower garden" sequence. While the compression rate decreases significantly in the "table tennis" sequence. The reason behind this is that these permutations distort the probability distribution of run-lengths, and render the Huffman tables used less than optimal. A possible direction for future research is to construct different Huffman tables optimal for different permutations. The Huffman tables, together with the corresponding permutation list, is used as the secret key to encrypt the images.

6 Conclusion

How to incorporate cryptographic technology with digital image processing technology to provide multimedia security does not seem to be considered in the previous literatures. The main contribution of this paper is the idea of incorporating cryptographic techniques (random algorithms) with digital image processing techniques (image compression algorithms) to achieve compression (decompression) and encryption (decryption) in one step. We illustrate this idea by applying traditional cryptographic techniques such as randomized permutation and probabilistic encryption to the MPEG video codec standard to provide a light-weight encryption mechanism.

Multimedia security is an interesting area. Our approach is just a beginning. More experimental studies are needed to investigate the cryptographic and compression property of the random permutation list and its effects on the compression rate. We hope our work would stimulate those codec experts to consider the option of encryption and decryption in the future codec design and apply our methods to other compression techniques such as wavelet.

Acknowledgements

The author would like to thank the anonymous referees of the ACM Multimedia'96 program committee to provide me many suggestions to improve the contents of this paper.

References

[1] Coded Representation of Picture, Audio and Multimedia/Hypermedia Information. Committee Draft of Standard ISO/IEC 11172-2, 1993.

[2] I. Agi and L. Gong. An Empirical Study of Secure MPEG Video Transmissions. In *The Internet So-*

ciety Symposium on Network and Distributed System Security, Feburary 1996.

[3] R. Durstenfeld. Algorithm 235: Random Permutation. *Communication of ACM*, 7(7):420, July 1964.

[4] D. Le Gall. MPEG: A Video Compression Standard for Multimedia Applications. *Communication of ACM*, 34(4):46–58, 1991.

[5] K. Gong and L.A. Rowe. Parallel MPEG-1 Video Encoding. In *Proceedings of the 1994 Picture Coding Symposium*, September 1994.

[6] D. A. Huffman. A Method for the Construction of Minimum Redundancy Codes. *Proceedings IRE*, 40:1098–1101, 1962.

[7] B. Macq and J. Quisquater. Cryptology for Digital TV Broadcasting. *Proceedings of The IEEE*, 83(6):944–957, June 1995.

[8] T.B. Maples and G.A. Spanos. Performance Study of a Selective Encryption Scheme for the Security of Networked, Real-time Video. In *Proceedings of 4th International Conference on Computer Communications and Network*, September 1995.

[9] J. Meyer and F. Gadegast. Security Mechanisms for Multimedia Data with the Example MPEG-1 Video. http://www.cs.tu-berlin.de/ phade/ phade/secmpeg.html, 1995.

[10] K. Patel, B. Smith, and L. Rowe. Performance of a Software MPEG Video Decoder. In *Proc. ACM Multimedia'93*, August 1993.

[11] W. Pennebaker and J. Mitchell. *JPEG Still Image Data Compression Standard*. Van Nostrand Reinhold, 1993.

[12] W. Pennebaker, J. Mitchell, and et. al. Arithmetic Coding Articles. *IBM Journal of Research and Development*, 332(6):717–774, 1988.

[13] G. Wallace. The JPEG Still Picture Compression Standard. *Communication of ACM*, 34(4):30–44, 1991.