

A Fast MPEG Video Encryption Algorithm

Changgui Shi Bharat Bhargava
Department of Computer Sciences
Purdue University
West Lafayette, IN 47906, USA
{shicg,bb}@cs.purdue.edu

Abstract

Multimedia data security is important for multimedia commerce. Previous cryptography studies have focused on text data. The encryption algorithms developed to secure text data may not be suitable to multimedia applications because of large data sizes and real time constraint. For multimedia applications, light weight encryption algorithms are attractive.

We present a novel MPEG Video Encryption Algorithm, called VEA. The basic idea of VEA is to use a secret key randomly changing the sign bits of all of the DCT coefficients of MPEG video. VEA's encryption effects are achieved by the IDCT during MPEG video decompression processing. VEA adds minimum overhead to MPEG codec, one bitwise XOR operation to each non zero DCT coefficient. A software implementation of VEA is fast enough to meet the real time requirement of MPEG video applications. Our experimental results show that VEA achieves satisfying results. We believe that it can be used to secure video-on-demand, video conferencing and video email applications.

Keywords: Multimedia data security, MPEG video encryption, DES, MPEG codec.

1 Introduction

Multimedia data security is important for multimedia commerce. For example, in video on demand and video conferencing applications, it is desirable that only those who have paid for the services can view their videos or movies.

One method to secure distributed multimedia applications is to use authentication control mechanisms. However, applying this method alone is not enough to secure multimedia data broadcast on wireless, satellite, or Mbone networks. Another method is to encrypt multimedia data, e.g., using DES(Data Encryption Standard)[17]. However, DES is very complicated and involves large computations. A software DES implementation is not fast enough to process the vast amount of data generated by multimedia applications and a hardware DES implementation(a set-top box)

adds extra costs both to broadcasters and to receivers.

The challenges of multimedia data encryption come from two facts. First, multimedia data sizes usually are very large(e.g., two hours MPEG-1 video has a size about 1GB). Second, multimedia data needs to be processed in real time(e.g., MPEG-1 requires data processing rate of 1.5MB/sec). Processing vast amount of data puts great burden on video codec, storage space requirements and network communications. Heavy weight encryption and decryption algorithms(during or after encoding phase) will aggravate the problem and increase the latency.

For some commercial applications, such as pay-per-view, very expensive attacks of the scrambled multimedia data are not interesting to adversaries[11], because most multimedia videos are different from military secrets or financial information. For multimedia applications, information rate is very high, but the information value is very low[11]. To break such encryption code is much more expensive than to buy the programs. Hence, light weight encryption and decryption algorithms are suitable to secure certain multimedia data.

Maples[12] and Tang[18] proposed some methods to encrypt MPEG videos. Their methods are either involving heavy computations[12] or decreasing compression rate[18].

In this paper, we present a novel light weight MPEG video encryption algorithm, called VEA. The basic idea of this algorithm is to use a secret key randomly changing the sign bits of all of the DCT coefficients. The encryption effects are achieved by the IDCT during MPEG video decompression processing. This algorithm adds minimum overhead to MPEG encoding and decoding, one XOR operation to each non zero DCT coefficient. A software implementation is fast enough to many multimedia applications, such as VOD systems and video conferencing systems.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 is a brief introduction of MPEG-1 compression model. Section 4 describes VEA MPEG video encryption algorithm. Section 5 analyzes VEA algorithm. Section 6 considers of VEA implementation issues. Section 7 shows some experimental results. Section 8 presents our discussions. Finally, section 9 is conclusions.

2 Related Work

In most multimedia data scrambling systems, data is treated as a bitstream and encrypted with DES[17] algorithm, such as vic[13] and [12]. In those systems, the encryption operations are processed after MPEG compression operations;

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

the decryption operations are processed before MPEG decompression operations. Those systems add latency to real time video delivering and involve heavy computations.

Tang[18] introduced some methods to incorporate MPEG compression with encryption in one step. Tang's methods use a random permutation list to replace the zig-zag order to map the DCT coefficients to a 1×64 vector. Since mapping according to the zig-zag order and mapping according to a random permutation list(secret key) have the same computational complexity, the encryption and decryption add very little overhead to the video compression and decompression processes. Tang's methods decrease video compression rate. The reason is that the random permutations distort the probability distribution of DCT coefficients and render the Huffman table used less than optimal. Our proposed algorithm, VEA, can also be incorporated with MPEG codec. But VEA does not decrease video compression rate.

In our previous work[16], we developed a light weight MPEG video encryption algorithm which incorporate encryption (decryption) with MPEG video compression (decompression). We use a permutation of the Huffman codeword list as a secret key. During MPEG coding (decoding), our encoder (decoder) uses the secret key instead of the standard Huffman codeword list. For those who do not have the secret key and use the standard Huffman codeword list to decode an encrypted MPEG video, they will not get the original video images.

Our previously developed algorithm has both an advantage and a disadvantage:

- The advantage is that no overhead is added to MPEG codec since the encryption/decryption processing does not cost extra computation.
- The disadvantage is the limitation of key space. First, since MPEG compression rate depends on the Huffman codeword list, if we use an arbitrarily permuted Huffman codeword list during a MPEG codec, we may decrease the compression rate. To avoid affecting compression rate, we limit the permutation of Huffman codeword list, the key generation, among those codewords which have the same length. Thus we actually reduce the key space. Second, it seems not all of permutations of the Huffman codeword list can be used as encryption keys. Different keys present different encryption results. This makes key generation difficulty. A key has to be tested before using.

The algorithm we present in this paper does not have this drawback.

3 Background

MPEG(Moving Picture Experts Group) standards, including MPEG-1, MPEG-2 and MPEG-4, are widely used in multimedia applications. MPEG standards consists of three parts: video, audio, and mechanisms to control interleaving streams. In here, we briefly describe MPEG-1 video codec. Similar operations can be found in MPEG-2 and MPEG-4 codec models.

In MPEG-1 video coding model[5, 6, 8], a video is composed of a sequence of group of pictures(GOPs). Each GOP is a series of I, P and B frames. I frames are coded using spatial compression independent of other frames. P frames are computed by using motion compensation prediction by coding the relative motion and noncompensated information

compared to the previous I and P frames. B frames are computed by using bidirectional motion compensation, relative to the previous and following I and P frames.

Each of I, P and B frames is divided into macroblocks. A macroblock is a 16×16 pixel array which is further subsampled into four 8×8 luminance Y blocks, one 8×8 chrominance Cr block and one 8×8 chrominance Cb block, as shown in Figure 1.

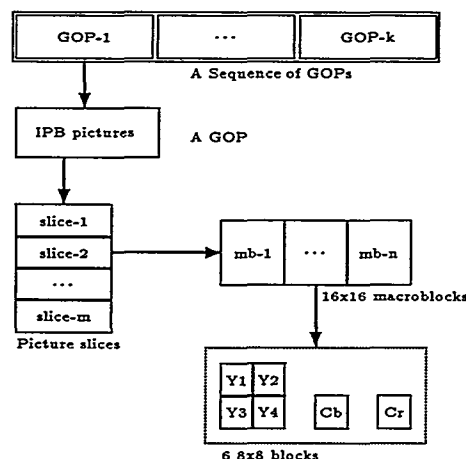


Figure 1: MPEG-1 video coding

Each of 8×8 Y, Cb and Cr blocks is fed to a pipeline of DCT, quantization and Huffman entropy coding, as shown in Figure 2. DCT concentrates most of the energy in the lower spatial frequencies, i.e., the upper-left corner of the 8×8 block. After quantization, many DCT coefficients become zeros. The quantization output is linearized according to the zig-zag order to a vector $\langle DC, AC_1, AC_2, \dots, AC_{63} \rangle$, as shown in Figure 3.

DC coefficient denotes the average brightness in the spatial block, and AC coefficients contain detail image information. Zig-zag sequencing of AC coefficients places high-energy, low-frequency coefficients before the low-energy, high-frequency coefficients.

The run length encoding turns vector $\langle DC, AC_1, AC_2, \dots, AC_{63} \rangle$ into a sequence of (skip, value) pairs, then the Huffman entropy coding is used to change the (skip, value) pair sequence into a compressed bitstream. MPEG standard provides a Huffman codeword table, where more frequently occurring (skip, value) pairs are assigned fewer bits in order to achieve high compression rate. Every Huffman codeword reserves a sign bit, 0 mean positive, 1 mean negative. These sign bits are the exact positions to encrypt the video in VEA.

4 Our Algorithm

MPEG video encryption aims to prevent unauthorized receivers from decoding the video programs by scrambling them. The general scheme is to apply an invertible transformation, E_{k_1} , to video stream, S , called *plaintext*, that produces a bitstream C , called *ciphertext*, $C = E_{k_1}(S)$. An authorized receiver, who has a secret key, k_2 , can decrypt

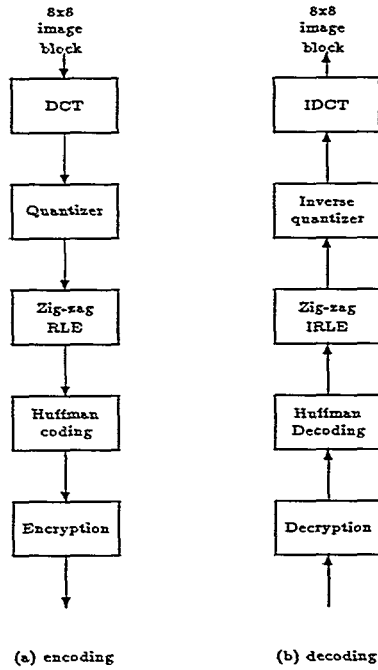


Figure 2: MPEG codec with cryptography

the video program by the transformation $D_{k_2} = E_{k_1}^{-1}$. The decryption operation is

$$D_{k_2}(C) = E_{k_1}^{-1}(C) = E_{k_1}^{-1}(E_{k_1}(S)) = S$$

where parameter k_1 is called encryption key, and k_2 is called decryption key.

Previous studies of cryptography have focused on text data. The encryption algorithms developed to secure text data, such as DES[17] and RSA[17], may not be suitable to multimedia applications because of large data size, low value and real-time constraint. Security is a trade-off between the cost of the data being protected and the cost it takes an attacker to get that data. Protecting data worth only several dollars with a security system that costs a million dollars to break is obviously a bad investment. The trade-off is to find the balance between the cost to an attacker and the value of the data to its owner. The cost to break a security system can be measured in many ways, which include the amount of computer time necessary to perform the security break, the amount of time and money spent by attackers. The costs of a multimedia security system include the amount of investment by data provider and the payment required for customer service. In multimedia applications, data providers, such as VOD providers, do not want to spend too much money on their security equipments; data receivers do not want to buy extra devices (hardware boxes, or co-processors). Multimedia applications appeal those encryption algorithms which require less computation and low implementation cost. For multimedia data, we only need to provide sufficient security level such that the cost to break the security system is much more expensive than to buy the

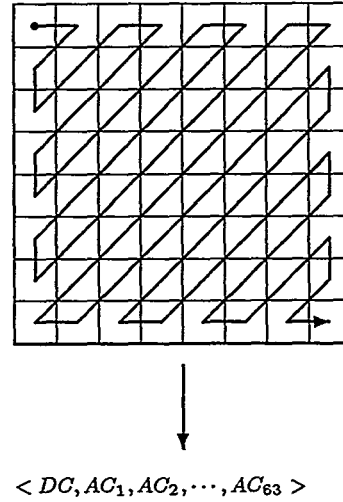


Figure 3: The zig-zag scan ordering

program. Hence, light-weight and cost-effective multimedia data encryption algorithms should be developed.

Due to the special property of MPEG video structure, it is not necessary to encrypt the video bit by bit. Our MPEG video encryption algorithm, VEA, is a selective encryption algorithm which only operates on the sign bits of DCT coefficients of a MPEG compressed video. VEA's secret key, k , is a randomly generated bitstream of length m which can be represented as $k = b_1 b_2 \dots b_m$. A MPEG compressed video S is a bitstream which can be represented as

$$S = \dots s_1 \dots s_2 \dots s_m \dots \\ s_{m+1} \dots s_{m+2} \dots s_{2m} \dots$$

where $s_i (i = 1, 2, \dots)$ are all of the sign bits of DC and AC coefficients (for DC coefficients of Y, Cr and Cb blocks of I-frames, these are the sign bits of the differential values, since those DC coefficients are differentially coded).

VEA's encryption function, E_k , can be described as

$$E_k(S) = \dots (b_1 \oplus s_1) \dots (b_m \oplus s_m) \dots \\ (b_1 \oplus s_{m+1}) \dots (b_m \oplus s_{2m}) \dots$$

where $\oplus$ is the binary XOR operation. The encryption operations randomly change the sign bits of DCT coefficients. According to the given secret key, a sign bit is either not changed (if the corresponding bit of the key is 0), or changed from positive to negative (0 to 1), or changed from negative to positive (1 to 0).

Pseudocode in Table 1 describes VEA encryption operations in detail. (Table 1 should not be considered as an optimal implementation of VEA, its only explains the algorithm). The procedure VEA takes a secret key and a MPEG video file as input parameters. Its output is a VEA encrypted MPEG video file.

In Table 1, we add resynchronization points at the beginning of each GOP (Group Of Picture). A resynchronization point is the position of MPEG video where VEA restarts to use its secret key (from the first bit). This is useful, because video transmission in networks is not reliable, noise

```

procedure VEA(
  int m,          /* key length */
  bit key[m],     /* secret key */
  char *mpeg_video, /* input file */
  char *vea_mpeg_video /* output file */
{
  int n;          /* buffer size */
  bit video[n];   /* input buffer */
  file in;        /* for input */
  file out;        /* for output */
  int k,l,i=0;
  in=open(mpeg_video,"r");
  out=open(vea_mpeg_video,"w");
  while(!eof(in)) {
    l=read(video, n, in); /* read l bits */
    for(k=0;k<l;k++){
      switch(video[k]) {
        case (beginning of a GOP):
          i=0; /* resynchronization */
          break;
        case (a sign bit of DCT coefficient or
              a sign bit of DC differential value
              of Y, Cr or Cb block of I frame):
          video[k]=video[k] xor key[i];
          i= ++i mod m;
      } /* end switch */
    } /* end for */
    write(video, l, out); /* write l bits */
  } /* end while */
  close(in); close(out);
} /* end procedure */

```

Table 1: VEA algorithm

and errors exist, dropping bits and even video frames can happen[2, 3, 4, 20]. VCR like functions[10] in VOD systems also require that VEA can start to decrypt video at many points, such as at the beginning of each I frame, or at the beginning of each GOP, or at the beginning of every 30 frames, etc. Resynchronization points allow users decrypting certain segments of a video. In a VEA implementation, one can decide his/her own resynchronization points.

VEA's encryption effects are achieved by the IDCT during MPEG video decoding. Even if only some of DCT coefficients are changed, these changes will propagate to most of IDCT coefficients.

For those who have the secret key, they can decrypt the video and get the original video. VEA's decryption function, E_k^{-1} is the same as its encryption function since $E_k(E_k(S)) = S$. In VEA, encryption key and decryption key are the same. For those who do not know the secret key, their decoders will play quite different images from the original video, because most image pixel values are changed.

5 Analysis

Suppose X is a 8×8 image block. After DCT transformation, we get a 8×8 Y block with

$$y_{kl} = \frac{c(k)c(l)}{4} \left[\sum_{i=0}^7 \sum_{j=0}^7 x_{ij} \cos \frac{(2i+1)k\pi}{16} \cos \frac{(2j+1)l\pi}{16} \right]$$

where

$$c(k) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } k=0; \\ 1 & \text{otherwise.} \end{cases}$$

When we apply VEA encryption operation to Y (for simplicity, we ignore the MPEG quantization, the zig-zag order and the Huffman entropy coding operations), we actually change the signs of elements of Y according to the given secret key. The encryption results, Z , can be represented as

$$z_{kl} = e_{kl} y_{kl} \quad k, l = 0, \dots, 7.$$

where

$$e_{kl} = \begin{cases} 1 & \text{if } y_{kl} \text{ sign is not changed;} \\ -1 & \text{if } y_{kl} \text{ sign is changed.} \end{cases}$$

As we know, the definition of forward IDCT is

$$x_{ij} = \sum_{k=0}^7 \sum_{l=0}^7 \left[\frac{c(k)c(l)}{4} y_{kl} \cos \frac{(2i+1)k\pi}{16} \cos \frac{(2j+1)l\pi}{16} \right]$$

If one does not decrypt Z before the IDCT computations of MPEG decoding (again for simplicity, we ignore the MPEG dequantization, the reverse zig-zag order and the Huffman entropy decoding operations), then his decoder will get image block $\hat{X}$ with

$$\hat{x}_{ij} = \sum_{k=0}^7 \sum_{l=0}^7 \left[\frac{c(k)c(l)}{4} z_{kl} \cos \frac{(2i+1)k\pi}{16} \cos \frac{(2j+1)l\pi}{16} \right]$$

$$= \sum_{k=0}^7 \sum_{l=0}^7 \left[\frac{c(k)c(l)}{4} e_{kl} y_{kl} \cos \frac{(2i+1)k\pi}{16} \cos \frac{(2j+1)l\pi}{16} \right]$$

$$= \sum_{k=0}^7 \sum_{l=0}^7 \left[\frac{c(k)c(l)}{4} y_{kl} \cos \frac{(2i+1)k\pi}{16} \cos \frac{(2j+1)l\pi}{16} \right] +$$

$$\sum_{k=0}^7 \sum_{l=0}^7 \left[\frac{c(k)c(l)}{4} (e_{kl} - 1) y_{kl} \cos \frac{(2i+1)k\pi}{16} \cos \frac{(2j+1)l\pi}{16} \right]$$

If we denote a 8×8 block N 's elements n_{ij} as

$$\sum_{k=0}^7 \sum_{l=0}^7 \left[\frac{c(k)c(l)}{4} (e_{kl} - 1) y_{kl} \cos \frac{(2i+1)k\pi}{16} \cos \frac{(2j+1)l\pi}{16} \right]$$

then

$$\hat{x}_{ij} = \tilde{x}_{ij} + n_{ij}$$

where n_{ij} is the noise or shift added to pixel x_{ij} and $\tilde{x}_{ij}$ is the pixel value a decoder (after decryption) can get. $\tilde{x}_{ij}$ is close to but not exactly the same as the original image pixel value x_{ij} because of the lossy property of MPEG compression processes.

To summarize, for 8×8 blocks of $X, Y, Z, \hat{X}, \tilde{X}$ and N , we have

$$\begin{aligned} Y &= DCT(X) \\ Z &= VEA(Y) \\ \hat{X} &= IDCT(VEA^{-1}(Z)) = IDCT(Y) \\ \tilde{X} &= IDCT(Z) = \hat{X} + N \end{aligned}$$

where $\hat{X}$ is the image one can get from decryption and decoding ($X \approx \hat{X}$), $\tilde{X}$ is the image that unauthorized receivers (without decryption) can get, and N is the noise or shift added to X by VEA encryption operations.

Above analysis shows that though VEA only changes some signs of DCT coefficients, most image pixel values will be changed during inverse DCT(IDCT) of MPEG decoding operations. The analysis also shows that VEA actually is an image cipher algorithm. It shifts image pixel values according to a given secret key.

For a MPEG compressed video, if it has n none zero DCT coefficients after quantization, then the computational complexity of VEA is $O(n)$, since it only does one XOR operation for each none zero DCT coefficient. Number n is dependent on each MPEG video. We measured a segment of "table tennis" video. The DCT coefficient sign bit rate is only about 13.8% percent of the video bitstream. VEA only operates on a small number of bits of the whole video bitstream, hence, it is fast.

VEA is much more efficient than DES algorithm because it only selectively encrypts a small number of bits of the MPEG compressed video, and a selected bit is only XORed one time with the corresponding bit of the secret key; while DES encrypts every bit of the compressed video, and each bit is XORed 8 times with the given secret key plus many transposition operations. To encrypt a MPEG video, a software implementation DES will cost much more time than a software implementation VEA does.

To attack a VEA encrypted MPEG video, for a key of length m , on the average, an adversary needs to try 2^{m-1} times in order to find a secret key. For example, if $m = 56$, then the difficulty to find a secret key is the same as to find a DES secret key, which is very difficult. Please note that the cost of obtaining a secret key is increased by MPEG decoding process which includes inverse DCT computations.

VEA has following properties:

1. One can encrypt a MPEG video many times, and decrypt the video in one time. Since for two keys, k_1 and k_2 ($k_1 \neq k_2$), VEA has a property of

$$E_{k_1}(E_{k_2}(S)) = E_{(k_1 \oplus k_2)}(S)$$

If denote $k_3 = k_1 \oplus k_2$, then one can use k_3 to decrypt the encrypted video in one time.

This property is useful, because in order to prevent adversary attack, one needs periodically change the secret key of a program. This property indicates that the cost of change a secret key with VEA is cheaper than the cost of change a secret key with DES, which has to do decryption first, than do encryption again.

2. In VEA, the length of a secret key is not limited. Using a long secret key will not increase encryption computations, but it will make adversaries more difficult to attack.

The *block encryption* algorithms developed to secure text data usually use short keys and complex computations, such as DES. This is because text data has small size and cryptography experts want to make adversaries reveal secret key computationally impossible even with the help of powerful computers.

For multimedia applications, we believe that the reverse is true. We should use long keys to prevent adversary attack and use simple computation algorithms

to pursue real time performance. Multimedia data already has large size. To reveal a secret key, adversaries need to spend much long time to do the computations.

Of course, using long keys makes key distribution difficult. A solution is to use a pseudorandom generator. The key distributed is a *seed* of the random number generator, the key actually used is generated by the pseudorandom generator. Thus, we turn our algorithm into a selective *stream encryption* algorithm. One can use an arbitrarily long key.

6 Implementation Consideration

6.1 Key Generation

VEA key generation is an easy task. Any uniformly distributed random number generator may work. The important thing is 1s and 0s should be uniformly distributed in the generated key. A long run of either 0s or 1s may not be a good VEA key. The former may leave many DCT coefficients not changed, the later is easy to be attacked.

VEA key distribution and storage management are challenging problems which are out of the scope of the discussions of this paper. One can consider to use (or modify) PGP (Pretty Good Privacy) [1] package or Kerberos [1] system to distribute and manage VEA secret keys. Since VEA, like DES, is a symmetric encryption algorithm, the methods used to distributed DES keys can also be used to distribute VEA secret keys. One can also use WWW browsers', such as *netscape's*, data encryption facilities to distribute VEA secret keys.

6.2 Variation

VEA implementation can vary from one key to many keys. VEA does not limit key length, it does not limit key numbers either. One can use different number of keys to a MPEG video encryption. For example, one can uses

1. one key to encrypt the whole video;
2. two keys, one for Y blocks, one for Cb and Cr blocks;
3. three keys, one for Y blocks, one for Cb blocks and one for Cr blocks; or
4. three keys, one for I frames, one for B frames and one for P frames.

Using multiple keys can increase transcodability of VEA compressed MPEG video. For example, if we use three separate keys, one for I frames, one for B frames and one for P frames, then when we dropped all B frames of the VEA encrypted MPEG video, the video still can be decrypted and decoded.

7 Experiments

We conducted our experiments on a Sun Sparc 10 station at the Raidlib of the Computer Science Department of Purdue University. We modified the Berkeley *mpeg-play* [14] and *mpeg-encode* [9] programs and used them to test VEA algorithm.

There were two purposes in our experiments:

1. to test the encryption results;
2. to find the overheads added to MPEG codec.

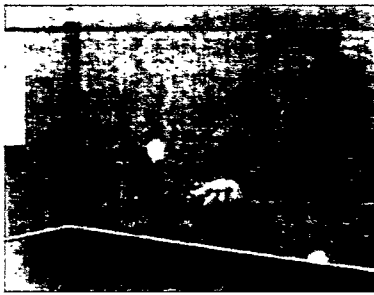


Figure 4: Original image



Figure 5: Encrypted AC coefficients

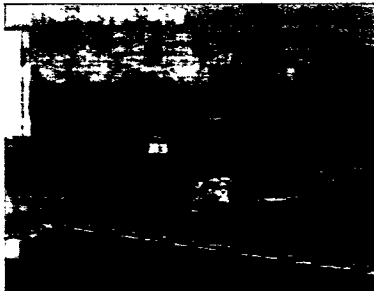


Figure 6: Encrypted ACs, DCs of Cr and Cb blocks

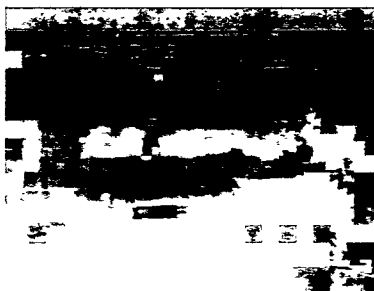


Figure 7: Encrypted all ACs and DCs

Test number	compression without encryption (fps)	compression and encryption (fps)
1	0.661813	0.672043
2	0.656599	0.657030
3	0.652742	0.678426
4	0.665779	0.692521
5	0.651042	0.659631
6	0.662691	0.665890
7	0.679348	0.679841
8	0.651042	0.673854
9	0.660939	0.665466
10	0.659631	0.678929
Average	0.660163	0.672363

Table 2: Overheads test results.

We selected key length of 128 bits. The input data is the "table tennis" MPEG-1 video clips. One of original clip is shown in Figure 4.

We conducted three kinds of experiments:

1. Encrypting all AC coefficients of DCT. One frame of the results is shown in Figure 5. The video image is blurred, but still comprehensible.
2. Encrypting all AC coefficients and DC coefficients of Cr and Cb blocks. One frame of the results is shown in Figure 6. The image is obscured, but still comprehensible.
3. Encrypting all DC and AC coefficients, including DCs of Y blocks. One frame of the results is shown in Figure 7. The image is incomprehensible.

Above three experiments represent three encryption levels. The low level encryption, which gets obscured video images, is also useful since video broadcasters do not always intend to prevent unauthorized receivers from receiving their programs, but rather intend to promote a contract with non-paying watchers.

From our experiments, we found that DC coefficients are more significant than the AC coefficients in an encryption. When we encrypted all DC coefficients of Y blocks, we always got satisfying encryption results.

We embedded VEA into the Berkeley *mpeg_encode*[9] program and used it to test the overheads added to MPEG codec. We conducted 10 tests each with different keys and encrypted all of DC and AC coefficients. All of the tests got incomprehensible video images. The time spent on compression only and compression with encryption is shown in Table 2 (where fps means frame per second).

From Table 2 we can see that relative time spent on the encryption is very small, only 1.81% of computation time. Hence, a software implementation of VEA is fast enough to secure some MPEG video applications.

8 Discussions

In this section, we consider VEA's advantages and disadvantages:

- Due to the well know MPEG structure and DCT coefficient distributions, a cryptanalysis expert may obtain obscure images by using a partial key. So VEA is not ideal for highly valuable multimedia applications. However, it is robust for those multimedia applications which the cost to break the encryption MPEG video is much higher than that to buy the video. VEA can be used to secure pay-per-view, to protect MPEG video

lectures, or to prevent children from watching adults movies, etc.

- VEA, as Tang's algorithm[18], is weak for plaintext attack. If an adversary knows the original video image(*plaintext*) and the encrypted video(*ciphertext*), for example, by subscribing to the program, he can easily reveal VEA secret key. So it is worthwhile to increase the VEA security level by frequently changing its secret key.
- When applied to MPEG video delivery, some algorithms encrypt every bit of the video bitstream, include the long run of zeros for resynchronization and the MPEG header bits. These encrypted long run of zeros and encrypted MPEG header bits can be used to reveal the secret key by adversaries [11]. VEA does not have this drawback since it does not encrypt the synchronization bits and MPEG header bits. Of course, deliberated VEA users will not encrypt the advertising part of their videos, since the advertise(*plaintext*) may appear else where which is easy to be obtained by adversaries.
- Since VEA does not change the MPEG video structure, a MPEG video encrypted with VEA is still readable to MPEG tools, while a MPEG video encrypted with DES is not readable to common MPEG tools. Hence, our encryption algorithm is weaker than DES in this aspect.

9 Conclusions

We have developed and tested a light weight MPEG video encryption algorithm, VEA. Our experiments have shown that VEA can achieved satisfaction MPEG video encryption results. We believe it can be used to secure many MPEG video applications. In fact, our algorithm can be extended to many other multimedia applications which uses DCT in their video compression processing, such as to secure JPEG pictures and DVD videos.

Considering practically implementation of VEA, one can incorporate encryption operations with MPEG video compression processing and incorporate decryption operations with MPEG video decompression processing. Or one can embeds VEA in VOD systems(such as *SBVS*[19]), or a video storage system(such as *symphony*[15]), video conferencing systems(such as *nv*[7]), or video email systems.

We are implementing a stand alone MPEG video encryption routine, called *mpeg_encrypt*, which takes a MPEG compressed video file name and a secret key as parameters. This routine can also be used as a decryption routine since decryption operation and encryption operation are the same in our algorithm.

Multimedia data security is challenging. We hope our algorithm would be useful in developing MPEG video applications.

Acknowledgement

This research was partially supported by a grant from NSF under NCR-9705931.

References

- [1] Derek Atkins, Paul Buis, Chris Hare, Robert Kelley, Carey Nachenberg, Anthony B. Nelson, Paul Phillips, Tim Ritchey, and William Steen. *Internet Security*. New Riders Publishing, Indianapolis, USA, 1996.
- [2] Bharat Bhargava, Shunge Li, Shalab Goel, Melliyal Annamalai, Puqi Tang, and Yihong Zhang. Impacts of Codec Schemes on Multimedia Communications. In *Proceedings of the International Conference on Multimedia Information Systems (MULTIMEDIA 96)*, New Delhi, India, pages 94-105. IETE, February 1996.
- [3] Bharat Bhargava, Shunge Li, Shalab Goel, and Jin Huai. A Distributed Video-on-Demand System for Video Conferencing. In *Proceedings of the International Conference on Multimedia Information Systems (MULTIMEDIA 96)*, New Delhi, India, pages 83-93. IETE, February 1996.
- [4] Bharat Bhargava, Shunge Li, Shalab Goel, Chunying Xie, and Changsheng Xu. Performance Studies for an Adaptable Video-Conferencing System. In *Proceedings of the International Conference on Multimedia Information Systems (MULTIMEDIA 96)*, New Delhi, India, pages 106-116. IETE, February 1996.
- [5] Vasudev Bhaskaran and Konstantinos Konstantinides. *Image and Video Compression Standards, Algorithms and Architectures*. Kluwer Academic Publishers, Norwell, Massachusetts, 1995.
- [6] John F. Koegel Buford. *Multimedia Systems*. ACM Press, Addison-Wesley Publishing Company, Inc., New York, 1994.
- [7] Ron. Frederick. Experiences with Real-time Software Video Compression. In *Proc. of the packet video workshop, Portland, Oregon September, 1994*.
- [8] D. Le Gall. MPEG: A Video Compression Standard for Multimedia Applications. *Communication of ACM*, 34(4):46-58, 1991.
- [9] Kevin L. Gong and Lawrence A. Rowe. Parallel MPEG-1 Video Encoding. In *Proceedings of the 1994 Picture Coding Symposium*, September 1994.
- [10] T.D.C. Little and D. Venkatesh. Prospects for Interactive Video-on-Demand. *IEEE, Multimedia* 1(3):14-24, Fall 1994.
- [11] B. Macq and J. Quisquater. Cryptology for Digital TV Broadcasting. *Proceedings of The IEEE*, 83(6):944-957, June 1995.
- [12] 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 The 4th International Conference on Computer Communications and Network*, September 1995.
- [13] Steven McCanne and Van Jacobson. *vic: A Flexible Framework for Packet Video*. In *Proc. of ACM Multimedia'95*, pages 511-522, San Francisco, California, Nov. 1995.
- [14] K. Patel, B. Smith, and L. Rowe. Performance of a Software MPEG Video Decoder. In *Proceedings of ACM Multimedia 93*, August 1993.

- [15] P. J. Shenoy, P. Goyal, S. Rao, and H. M. Vin. Symphony: An Integrated Multimedia File System. In *Proceedings of SPIE/ACM Conference on Multimedia Computing and Networking (MMCN98)*, San Jose, CA, pages 124-138, January 1998.
- [16] C. Shi and B. Bhargava. Light-weight MPEG Video Encryption Algorithm. In *Proc. of the Int'l Conf. on Multimedia, (Multimedia98, Shaping The Future)* January 23 - 25, 1998, pages 55-61, New Delhi, India. IETE, Tata Mcgraw-Hill Publishing Company.
- [17] Douglas R. Stinson. *Cryptography, Theory and Practice*. CRC Press, Inc., New York, 1995.
- [18] Lei Tang. Methods for Encrypting and Decrypting MPEG Video Data Efficiently. In *Proceedings ACM Multimedia96*, pages 219-229, Boston, MA., November 1996.
- [19] M. Vernick, C. Vemkatramani, and T. Chiueh. Adventures in Building the Stony Brook Video Server. In Wendy Hall and T. D. C. Little, editors, *Proceedings ACM Multimedia 96*, Boston, MA, November 1996. Addison-Wesley Publishing Company, Inc.
- [20] W. Zeng and B. Liu. Geometric-structure-based directional filter for error concealment in image/video transmission. *Proc. SPIE Wireless Data Transmission at Information Systems/Photonics East'95*, vol 2601, pp. 145-156, Oct. 1995.