MPEG Video Encryption in Real-time Using Secret Key Cryptography*

Changgui Shi Sheng-Yih Wang Bharat Bhargava
Department of Computer Science
Purdue University
West Lafayette, IN 47906, USA.

Abstract

We present a fast MPEG video encryption algorithm called RVEA which encrypts selected sign bits of the DCT coefficients and motion vectors using secret key cryptography algorithms such as DES or IDEA. RVEA features bounded computation time for any size of video frame and is robust to both plaintext and ciphertext attack. Since it adds a very small overhead to the MPEG video compression process, a software implementation is fast enough to meet the real-time requirement of MPEG video applications.

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

1. Introduction

Multimedia data security is very important for commercial multimedia applications. For example, in commercial video-on-demand applications, it is important to assure that only those who have paid for the services can view the videos. Similarly, in business video conferencing applications, it is imperative that only authorized persons can join the conference and share the video data.

One way to secure distributed multimedia applications is to encrypt multimedia data using secret key cryptography algorithms such as DES (Data Encryption Standard) or IDEA (International Data Encryption Algorithm) [12]. These algorithms, however, involve complicated computations. Software implementations of these algorithms are not fast enough to process the vast amount of data generated by multimedia applications. Hardware implementations will add extra costs to both video providers and video receivers.

The challenges of multimedia data encryption come from two facts. First, the size of multimedia data is usually very large. For example, the data size of a two-hour MPEG-1 video is about 1 GB. Second, multimedia data need to be processed in real-time. Processing huge volume of data puts a great burden on video codec, storage systems, and network communications. Heavy-weight encryption and decryption algorithms (during or after the encoding phase) will aggravate the problem and increase the latency.

For some commercial video applications such as pay-per-view program, information rate is very high, but information value is very low [5]. Very expensive attacks are not interesting to adversaries [5] because it is much more expensive to break such encryption code than to buy the programs. Hence, light-weight encryption algorithms which can provide sufficient security level and have an acceptable computation cost are attractive to MPEG video applications.

In this paper, we present an efficient MPEG video encryption algorithm called RVEA (Real-time Video Encryption Algorithm). RVEA is based upon our previous work, the VEA [11][10] algorithm, with the following improvements: First, the security of RVEA is significantly improved by adopting secret key cryptography algorithms to encrypt the data. Second, RVEA reduces and bounds its computation time by limiting the maximum number of bits selected.

Even after adding all the improvements described above, this algorithm still adds very small overheads to the MPEG video compression process. A software implementation is fast enough to meet the real-time requirement of MPEG video applications.

The rest of this paper is organized as follows.

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

Section 2 is a brief introduction to MPEG video compression model. Section 3 discusses related work. Section 4 presents RVEA in details. Section 5 analyzes the algorithm. Section 6 shows some experimental results. Finally, discussion and conclusion are given in section 7.

2. Background

An MPEG-1 video [2] is composed of a sequence of groups of pictures (GOPs). Each GOP is composed of a series of I, P, and B pictures. I pictures are intraframe coded without any reference to other pictures; P pictures are predictively coded using a previous I or P picture; and B pictures are bidirectionally interpolated from both the previous and following I and/or P pictures.

Each picture is divided into macroblocks. A macroblock is a 16×16 pixel array. Macroblocks belonging to I pictures are spatially encoded. Macroblocks belonging to B and P pictures are temporally interpolated from the corresponding reference picture(s), and the difference between the actual and reference values is encoded. The interpolation process also produces up to two motion vectors, a forward prediction vector and a backward prediction vector, for each macroblock in the reference pictures. Regardless of the type of pictures it belongs to, each macroblock is further subsampled into four 8×8 luminance blocks (Y blocks), two 8×8 chrominance blocks (a Cr block and a Cb block), as shown in Figure 1.

Each of the 8×8 Y, Cb, and Cr blocks is fed to a pipeline of DCT (Discrete Cosine Transformation), quantization and Huffman entropy coding, as shown in Figure 1. 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 zero. The quantization output is linearized in the zig-zag order to a vector $< DC, AC_1, AC_2, \cdots, AC_{63} >$. DC coefficient denotes the average brightness in the spatial block, and AC coefficients contain detail image information. Both the DC coefficients of I blocks and the motion vectors of B and P macroblocks are differentially coded.

The run length encoding (RLE) turns a vector $< AC_1, AC_2, \cdots, AC_{63} >$ 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. The MPEG standard provides a default Huffman codeword table which assign fewer bits to more frequently occurring values to achieve a higher compression ratio. Every Huffman codeword reserves a sign bit(0 mean positive and 1 mean negative). These **sign bits** are the exact positions we encrypt the video.

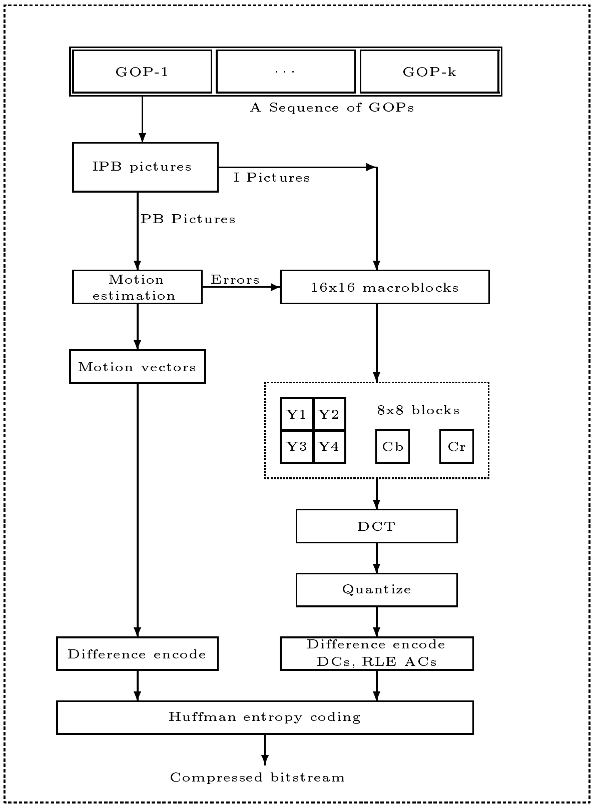

Figure 1. MPEG-1 video coding.

3. Related Work

A straightforward way to secure MPEG video is to treat the video as a bitstream and encrypt it with the DES. However, a software DES implementation is too slow to meet the real-time requirement of MPEG video playback. One strategy to solve this problem is to selectively encrypt only portions of MPEG video [4, 6].

One selection scheme proposed in the literature is to encrypt only MPEG video headers (including GOP headers, slice headers and macroblock headers). However, this scheme is not effective against attack due to the following reasons. First, the headers contains mostly standard information. Attackers can easily guess the information. Second, in many MPEG applications, a video stream is indexed by frame in order to perform synchronization. Therefore, the beginning of each frame is known [8]. Third, when block encrypted video headers are sent over a noisy channel, bit errors occurred in an encrypted block will propagate to the whole block when decrypting the video. It can make a decoder lose synchronization during a video playback.

Another scheme to selectively encrypt the video is to encrypt I frames only. One may think that P and B frames are useless without knowing the corresponding reference I frames. But great portions of the video could be visible because some of the P and B frames may contain intra-coded I blocks. Therefore, encrypting only I frames does not provide a satisfactory security level [1].

Other selective MPEG video encryption methods include (1) encrypting all headers plus DCs and lower AC terms [7]; (2) encrypting one half of a frame with DES/IDEA, and the other half of the frame with "one-time-pad" generated from that frame[8]. In those systems, the encryption operations are processed after MPEG compression operations; the decryption operations are processed before MPEG decompression operations. Those systems add latency to real-time video delivering and overhead to video decoding.

Tang [13] 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 (the secret key) have the same computational complexity, the encryption and decryption add very little overhead to the video compression and decompression processes. However, Tang's methods decrease the 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.

In our previous work [10, 11], we have developed fast MPEG video encryption algorithms which use a secret key randomly changing the sign bits of all DCT coefficients and the sign bits of motion vectors. These algorithms can achieve satisfying encryption results with less computations. However, they are very weak in plaintext attack. If an adversary has several frames of plaintext and the whole ciphertext, he can easily find secret key and decrypt the whole video. In this paper, we present a new algorithm, called RVEA, which is robust under both plaintext and ciphertext attacks.

4. RVEA

MPEG video encryption algorithms aim to prevent unauthorized receivers from decoding the video data by scrambling them. The general scheme is to apply an invertible transformation E_k to video stream S (the *plaintext*) and produces a bitstream $C = E_k(S)$ (the *ciphertext*). An authorized receiver who knows the secret key k can decrypt the video data by the transformation $D_k = E_k^{-1}$. RVEA is a selective encryption algorithm which operates only on the sign bits of both DCT coefficients and motion vectors of a MPEG compressed video. RVEA can use any secret key cryptography algorithms (such as DES or IDEA) to encrypt those selected sign bits.

In order to illustrate the operations of RVEA, if we ignore the other bits and concentrate only on the sign bits of an MPEG video stream, then an MPEG video S can be represented as

$$S = \quad \cdots s_1 \cdots s_2 \cdots s_m \cdots$$

where s_i $(i = 1, 2, \cdots)$ are the sign bits of motion vectors or the sign bits of DCT coefficients (for DC coefficients of I frames and motion vectors, these sign bits are the sign bits of the differential values, since they are differentially coded).

RVEA's encryption function, E_k (E is DES or IDEA), encrypts MPEG video slice by slice, as shown in Table 1. RVEA randomly changes the selected sign bits of DCT coefficients and sign bits of motion vectors. According to the given secret key, a sign is either not changed, or changed to its opposite sign.

RVEA selects at most 64 sign bits (8 bytes) from each macroblock. Let us denote an MPEG compressed 8×8 block as a bitstream of $\beta\alpha_1\alpha_2\alpha_3 \dots \alpha_n$ where β is the code of DC coefficient (if this block belongs to an I frame, then β is the code of the differential value, since the DC components of I frames are differentially encoded), α_i is the code for ith nonzero AC coefficient.

Each 16×16 macroblock contains six 8×8 blocks. They are Y1, Y2, Y3, Y4, Cr, and Cb. Figure 2 defines the order which RVEA selects the sign bits of β and α_i from each macroblock. The reason to select sign bits in this way is that DC coefficients are more significant than AC coefficients and lower frequency ACs are more significant than higher frequency ACs. RVEA achieve the goal of reducing and bounding its computation time by limiting the maximum number of bits selected.

Table 1. RVEA Algorithm.

```
Function RVEA:
{
  for (each macroblock M in a video slice){
    switch(M) {
       case (belong to a I frame):
          select at most 64 DC and AC
          sign bits in the order
          given in Figure 2;
          break;
       case (belong to a P frame):
          first select the sign bits of forward
          motion vector (f_x, f_y);
          then select at most 62 DC and AC
          sign bits in the order
          given in Figure 2;
          break;
       case (belong to a B frame):
          first select the sign bits of forward
          motion vector (f_x, f_y) and backward
          motion vector (b_x, b_y);
          then select at most 60 DC and AC
          sign bits in the order
          given in Figure 2;
       } /* end switch */
  } /* end for */
  encrypt the selected bits with DES or IDEA;
  put the encrypted bits to their
  original positions;
} /* end function */
```

RVEA's decryption operations are the same as its encryption operations.

5. Analysis

RVEA's encryption effects are achieved by the IDCT (inverse DCT) during MPEG video decoding. Although only some of the DCT coefficients are changed, these changes will propagate to most of the IDCT coefficients [11]. If an adversary does not know the secret key, his decoder will play pictures quite different from the original video frames because most image pixel values are changed.

Suppose X is a 8×8 image block. After DCT transformation, we get a 8×8 Y block, $Y = DCT(X)$. When we apply RVEA 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 elements of Y according to the given secret key. The encryption results, Z, can be represented as $Z = RVEA(Y)$. A decoder (after decryption) can

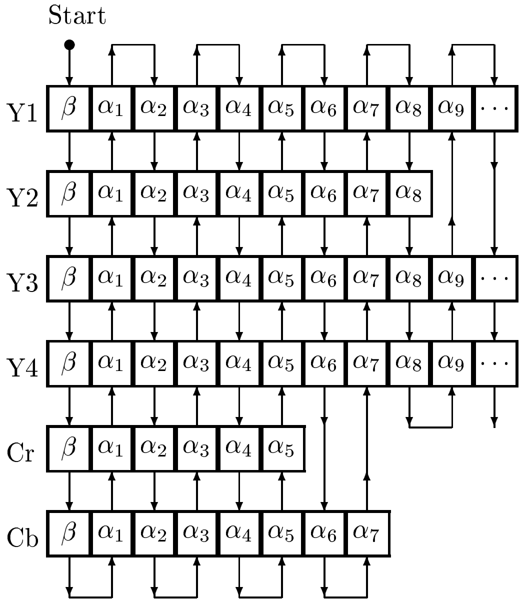

Figure 2. Bits selection order

get an image block $\tilde{X}$,

$$\tilde{X} = IDCT(RVEA^{-1}(Z)) = IDCT(Y).$$

$\tilde{X}$ is close to but not exactly the same as the original image pixel value X because of the lossy property of MPEG compression processes.

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 $\hat{X}$,

$$\hat{X} = IDCT(Z) = \tilde{X} + N$$

where N is the noise or shift added to pixel X. For 8x8 matrix A, if we define

$$|A| = \sum_{i=0}^{7} \sum_{j=0}^{7} |a_{ij}|$$

then

$$|\hat{X} - \tilde{X}| = |N|.$$

The value of $|N|$ will vary with X and the secret key used by RVEA.

The analysis above shows that although RVEA only changes some sign bits of DCT coefficients, most image pixel values will be changed during

IDCT of MPEG decoding operations. The analysis also shows that RVEA actually is an image cipher algorithm. It is the IDCT that shifts all image pixel values and propagates noise added by RVEA according to the given secret key.

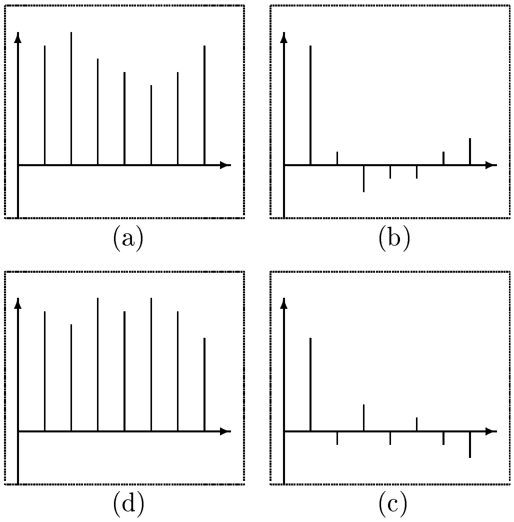

Figure 3. Encryption of differential values of DCs of I frames. (a) Signal values; (b) Differential values; (c) Encrypted differential values; (d) Result of decoding without decryption.

Changing the sign bits of DC coefficients of I frames can lead to a very dramatic change in the decoded frame. Since DC coefficients of I frames are differentially encoded, changing the sign bit of a differential value will affect all of its following DC coefficients during MPEG decoding. As an example, Figure 3(a) is a plot of the sequence of DC values of a I frame. Figure 3(b) shows the differential values of those DC values. RVEA encryption randomly changes the signs of these encoded values, as shown in Figure 3(c). Without a correct secret key, a decoder will get wrong DC values, as shown in Figure 3(d).

The differential encoding of DC coefficients and motion vectors in MPEG compression increases the difficulty of breaking RVEA encrypted videos. If the initial guess of a DC coefficient is wrong, it is very difficult to guess the following DC values correctly.

Changing the sign bits of motion vectors of P and B frames will mislead the adversary's decoder during video playback. When we change the sign bits of differential values of motion vectors, we actually change the directions of motion vectors. Because motion vectors are differentially encoded, the changes will also affect the magnitudes of motion vectors during decoding. Figure 4 illustrates the scenario for a P frame. Similar figure can be drawn for B frames. When decoding P and B frames, RVEA misleads the adversary's decoder to refer to wrong reference macroblocks.

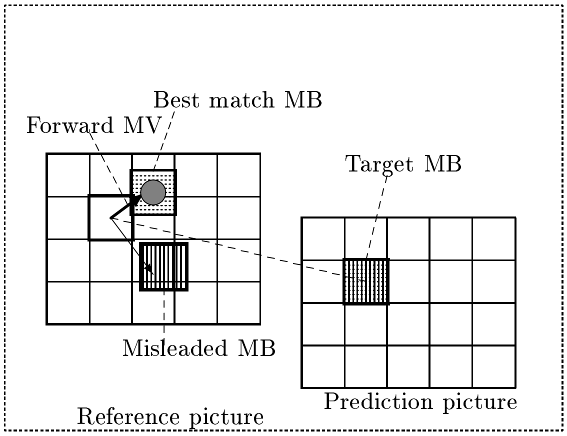

Figure 4. RVEA misleads adversaries' decoders to refer to wrong macroblock.

Theoretically, the difficulty of breaking an RVEA encrypted MPEG video is the same as the difficulty of breaking the underlining secret key cryptography algorithms. Even if plaintext and ciphertext are known, currently there is no practical method to determine the secret key in DES or IDEA. Please note that the cost of plaintext attack to RVEA is increased by the MPEG decoding process, which includes expensive IDCT computations.

Ciphertext attacks on RVEA encrypted MPEG videos are not practical either. Ciphertext attacks on RVEA can be done by trying all possible combinations of the sign bits of a frame and check if any one of the combination of the sign bits can produce a comprehensible video frame. The decorrelative property of AC coefficients of DCT makes a correlation attack not very easy. Even with a powerful computer, an automatic ciphertext attack (revealing secret key or generating quality pictures without human involvement) is hard, because the computer does not know if a guessed picture is comprehensible to human beings. A ciphertext attack with human involvement by playing picture puzzles of flipping the sign bits may get some blurred pictures or figure out some objects in certain frames. It will take many hours for an attacker to get a

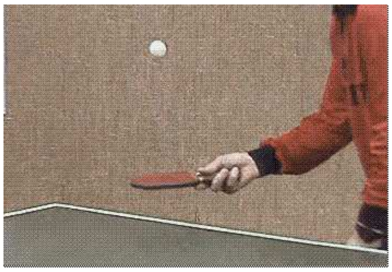

Figure 5. Original image frame.

Figure 6. Encrypted motion vectors only.

blurred picture. Anyone who found the picture interesting would rather buy the video than attack the ciphered video because the cost of buying the program is actually cheaper.

6 Experiments

To test RVEA encryption results, we embedded RVEA into the Berkeley *mpeg_encode* [3] program. We used IDEA in our RVEA implementation because IDEA is faster than DES. IDEA uses a key of 128 bits to encrypt a plaintext block of 64 bits. IDEA is generally considered to be very secure. No practical attack on it has been published.

The input data is the "table tennis" MPEG-1 video clips. One frame of the original clip is shown in Figure 5. Figure 6 shows the effect of encrypting only motion vectors of a B frame. Figure 7 shows one of the RVEA encrypted frames. All pictures in our RVEA tests are incomprehensible.

We found that the time spent on encryption with RVEA during the video compression process is only 2.55% of the total computation time. Hence, a software implementation of RVEA is fast enough to secure MPEG video in real-time MPEG applications.

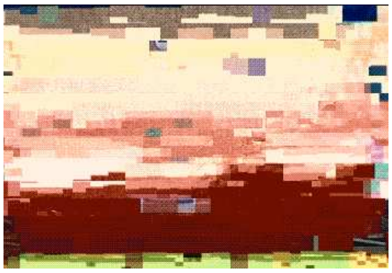

Figure 7. A RVEA encrypted frame.

7 Discussion and Conclusion

RVEA only selectively encrypts a fraction of the whole video. It is faster than encrypting the whole video with DES/IDEA. We found that in typical MPEG videos sign bits occupy less than 10% of a whole video bitstream. Therefore RVEA can save 90% of encryption time comparing to algorithms which encrypt the whole video.

RVEA encrypts at most 64 bits (8 bytes) for each macroblock. Thus, it considerably reduces encryption computations. For example, for 320×240 video frames, there are 20×15 macroblocks in each frame. To process data at 30 frames per second, RVEA needs to encrypt data at $20 \times 15 \times 30 \times 8$ bytes per second, which is about 72 KB/sec. Using IDEA, even a 66 MHz 486 PC can encrypt data at 300 KB/sec [9].

We do realized that applying RVEA encryption to the unit of video slice implies that the decryption can start only after the full slice is available; and the decryption followed by sign correction may impose certain amount of delay. Because the time spent in encryption computations is much longer than the communication delay, RVEA is still much faster than the approaches which encrypt the whole video frames.

To guarantee QoS (Quality of Service) in real-time video applications, it is desirable that the encryption/decryption time be bounded by a constant. Encryption and decryption should not take too much time, otherwise the video session will suffer from jitters. This is especially true for MPEG video applications because MPEG video frame sizes vary in time.

The encryption time spent by the algorithms which encrypt whole frames may vary with frame sizes. Larger frame size requires longer encryption time, which means longer delay. It may either degrade the quality of video playback or add extra

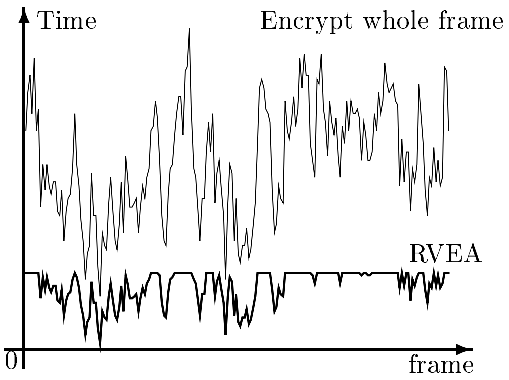

Figure 8. RVEA encryption time for each frame is bounded.

burden to the video frame synchronization.

Burstiness of data does not affect RVEA's encryption speed. RVEA encrypts at most 64 bits (8 bytes) of data for each macroblock, no matter how large the frame size is and what type (I,P,or B) the frame is (see Figure 8). This is a very desirable property to video decoding.

We have developed and tested a real-time MPEG video encryption algorithm, RVEA. RVEA can achieve satisfactory encryption results with less computations. A software RVEA implementation is fast enough to meet the real-time requirement of MPEG decoding. We believe that RVEA can be used to secure video-on-demand applications and pay-per-view programs. It is also possible to extend the idea of RVEA to non-MPEG compression scheme such as **H.263** for video-conferencing applications.

References

[1] I. Agi and L. Gong. An Empirical Study of MPEG Video Transmission. *In Proceedings of the Internet Society Symposium on Network and Distributed Systems Security, pp137-144. San Diego, CA.*, Feb. 1996.

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

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

[4] Y. Li, Z. Chen, S. Tan, and R. Campbell. Security Enhanced MPEG Player. *In proceedings of IEEE First International Workshop on Multimedia Software Development (MMSD 96), Berlin, Germany*, March 1996.

[5] B. Macq and J. Quisquater. Cryptoloy for Digital TV Broadcasting. *Proceedings of the IEEE, Vol. 83(6):944-957*, June 1995.

[6] 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 Neworks*, September 1995.

[7] J. Meyer and F. Gadegast. Security Mechanism of Multimedia Data with the Example MPEG-1 Video. *Available on WWW via http://www.powerweb.de/phade/phade.html*, 1995.

[8] Lintian Qiao and Klara Nahrstedt. Comparison of MPEG Encryption Algorithms. International Journal on Computers&Graphics*, Special Issue: "Data Security in Image Communication and Network", Vol. 22, No. 3, published bimonthly by Permagon Publisher*, January 1998.

[9] Bruce Schneier. *Applied Cryptography.* John Wiley & sons, INC., New York, 1996.

[10] C. Shi and B. Bhargava. An Efficient MPEG Video Encryption Algorithm. *In Proceedings of the 17th IEEE Symposium on Reliable Distributed Systems, West Lafayette, Indiana, USA. published by IEEE Computer Society, pp381-386. Los Alamitos, California*, Oct. 1998.

[11] C. Shi and B. Bhargava. A Fast MPEG Video Encryption Algorithm. *In Proceedings of the 6th ACM International Multimedia Conference, Bristol, UK*, pages 81–88, September 1998.

[12] Douglas R. Stinson. *Cryptography Theory and Practice.* CRC Press, Inc., New York, 1995.

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