An Efficient Encryption Scheme for MPEG Video

In-Kuan Cheong, Yi-Chin Huang, Yi-Shin Tung, Shian-Ru Ke, Wen-Chin Chen

Communication and Multimedia Lab.,

Department of Computer Science and Information Engineering,

National Taiwan University, Taipei, Taiwan

E-Mail: {ckuan, tung, yichin, kool, wcchen}@cmlab.csie.ntu.edu.tw

Abstract

In this paper, we propose an encryption scheme for MPEG video, which uses a permuted Huffman table to encode the input symbols, and then rotates and XORs (exclusive or) the encoded bit stream. This scheme adds very little computational overhead to the MPEG video compression process, and therefore the implementation is fast enough to meet the real-time requirement of MPEG video applications. In our analysis, it is also robust to both plaintext and ciphertext attacks. In addition, our scheme can be applied to all MPEG compression standards.

1. Introduction

MPEG (ISO Moving Picture Expert Group) is a compression standard for video processing and is widely used in multimedia applications, for example, VCD, DVD and VOD (Video-on-Demand). When the MPEG streams are transmitted over the public Internet, security has become very important because of Confidentiality and Copyright protection. Therefore, it is necessary to design special encryption algorithms for MPEG to ensure the security. The MPEG video encryption algorithms should aim towards efficient and real-time processing so that they can become an integral part of the video delivery process and at the same time preserve the highest security level and compression ratio.

The first intuitional solution is to encrypt multimedia data using secret key cryptography algorithms such as DES (Data Encryption Standard) or IDEA (International Data Encryption Algorithm). However, these algorithms involve complicated computations and the large size of multimedia data requires consuming a considerable amount of computation power. Moreover, the encryption/decryption speed is especially critical to multimedia data because real-time processing is often required.

More recently, several encryption algorithms applied to MPEG video streams emerged. Tang proposed the Signal Scrambling scheme [1], which consists of relatively simple permutations and/or scrambles the transformation coefficients. The speed of Signal Scrambling is very fast but it reveals serious security problems. Next, several researchers [2-4] used Selective Encryption algorithm to encrypt just some important portions of MPEG layered syntactic structures, such as header information, Intra-macroblocks, and motion vectors, etc. However, portions of the video are still visible and identifying the features introduces some computational overhead. The MHT scheme proposed by C.-P. Wu and C. C. Kuo [5][6] chooses several different Huffman tables from a vast number of possible candidates, and uses them alternatively to encode multimedia data. The MHT scheme requires very little computational overhead, but it is vulnerable to chosen-plaintext attack. Also in the MHT scheme, the security is in direct proportion to the number of chosen tables.

It means that the more tables are chosen the more secure it will be. Therefore, a considerable amount of memory will be used to work with MHT if higher security is desired.

In this paper, we propose an encryption scheme which is fast enough to meet the real-time requirement of MPEG video applications and also robust to both plaintext and ciphertext attacks. Furthermore, it only uses a minimal amount of memory.

2. High Performance, Low Complexity Encryption Scheme

In our proposal, we use three operations to fulfill encryption application requirements, namely security, speed and compression ratio. These operations are Huffman Tree Mutation, Byte Rotation and Bitwise XOR. Huffman Tree Mutation generates a different Huffman coding table for symbol encoding. Byte Rotation scrambles the bitstream, and thus the attackers cannot synchronize the plaintext with the ciphertext. Bitwise XOR is a very simple and fast operation. It can re-enforce the security. In order to operate these three components, the secret (user) key is divided into 3 parts, one for each operation.

Our scheme can be described by the following 9 steps and is shown in Figure 1.

1. Use the first part of user key k_1 to generate a permuted Huffman coding table T_1 by tree mutation process (based on the MHT scheme [5][6]).
2. Replace the original Huffman table with the new Huffman table T_1 in the encoder, such that the output bitstream will be encrypted by T_1 .
3. Use the second part of user key k_2 as a seed to generate a random vector of length n , $R_{si} = (r_{si0}, r_{si1}, \dots, r_{sin})$, where each r_{sij} is a 3-bit integer varying from 0 to 7.
4. Use the third part of user key k_3 as a seed to generate a random pattern of length m , $B_{si} = b_{si0}b_{si1} \dots b_{sim}$, where each b_{sij} is a 1-bit integer.
5. For a new symbol to be encoded, use the VLC encoder (with table T_1) to output its corresponding bit pattern.
6. According to vector R_{si} to rotate each output byte of the i -th segment of the encoded bitstream.
7. Use pattern B_{si} to XOR the i -th segment of the rotated bitstream.
8. If no more symbol is going to be encoded, end the process.
9. Before encoding the next symbol, the counter updates to reflect the encoded bit number or the length of the encoded period. If the counter overflows, go to step 3, otherwise go to step 5.

Figure 1 : The structure of our scheme

3. Analysis

Under a ciphertext-only attack, without any knowledge of the plaintext, the most reasonable approach is the brute-force exhaustive key search attack. The strength of resisting this attack relies on a large key space. In our scheme, the user key is divided into 3 parts (one for tree mutation, one for rotation and one for XOR). A Huffman coding table with m entries could be mutated into 2^m different tables, from which one is chosen in the first step of our scheme. Then in the third and the fourth steps, we use two k -bit integers as the seeds to generate random sequences. As a result, the size of the key space is

$$\text{Size (key-space)} = 2^m \times 2^k \times 2^k = 2^{(m+2k)}$$

In our implementation the key space is 2^{253} , it is practically impossible to perform a brute-force search in a key-space of this size.

Under a known-plaintext attack, because random sequences for rotation and XOR will change periodically when the counter overflows, our scheme is immune from this attack model. Under a chosen-plaintext attack, since we rotate each byte of the encoded bitstream, the attacker cannot synchronize the plaintext with the ciphertext. The case is just like the ciphertext-only attack, the attacker must try the exhaustive key search. Therefore, our scheme is secure under these 3 attack models.

The size of the encrypted bitstream will be the same as that of the original coding because our scheme replacing the Huffman table will not change the symbol coded length. This property retains the coding efficiency of compression systems.

4. Experimental Results

This section provides some simulation results of our scheme. Table 1 shows the percentage of time increase using our scheme compared to normal compression process.

No	Video Name	Percentage of time increase (encoder)	Percentage of time increase (decoder)
1	Missa.yuv	0.12 %	3.53 %
2	Grandma.yuv	0.65 %	3.55 %
3	Dancer.yuv	0.38 %	0.43 %
4	Stefan.yuv	0.30 %	2.29 %
5	Carphone.yuv	0.35 %	0.63 %
6	Container.yuv	0.89 %	1.63 %
7	Football.yuv	0.48 %	2.35 %
8	Singer.yuv	0.80 %	2.18 %

Table 1

As shown in Table 1, the execution time of the encoder with the encryption scheme increases less than 1%. On the other hand, the decoder using the decrypting process adds less than 5 % time. Therefore, the overall speed of our scheme is approximately the same as that of the normal compression process without encryption.

5. Conclusion

In this paper, we proposed a new scheme, which involves three simple operations: Huffman Tree Mutation, Byte Rotation and Bitwise XOR. Our scheme provides very efficient, fast and secure encryptions. Since our scheme does not change the architecture of the MPEG compressing process, it can be applied to all MPEG standards and added in current encoders/decoders easily.

6. Reference

- [1] L. Tang, "Methods for Encrypting and Decrypting MPEG Video Data Efficiently", Proceedings of the 4th ACM International Conference on Multimedia, pp. 219-229, November 1996.
- [2] L. Qiao and K. Nahrstedt, "Comparison of mpeg encryption algorithms", International Journal on Computers and Graphics, Special Issue on Data Security in Image Communication and Network, vol. 22, January 1998.
- [3] L. Qiao and K. Nahrstedt, "A New Algorithm for MPEG Video Encryption", Proceedings of the First International Conference on Imaging Science, Systems, and Technology, July 1997.
- [4] C. Shi, S.-Y. Wang, and B. Bhargava, "Mpeg Video Encryption in Real-time Using Secret Key Cryptography", Proc. of PDPTA'99, 1999.
- [5] C.-P. Wu and C. C. Jay Kuo, "Efficient Multimedia Encryption via Entropy Codec Design", Proceedings of SPIE International Symposium on Electronic imaging 2001, (San Jose, CA, USA), Vol. 4314, Jan. 2001.
- [6] C.-P. Wu and C. C. Jay Kuo, "Fast Encryption Methods for Audiovisual Data Confidentiality", Proceedings of SPIE International Symposia on Information Technologies 2000, (Boston, MA, USA), Vol. 4209, pp 284-295, Nov. 2000.