ON THE SECURITY OF MULTIMEDIA ENCRYPTION SCHEMES BASED ON MULTIPLE HUFFMAN TABLE (MHT)

Jiantao Zhou, Zhiqin Liang, Yan Chen, and Oscar Au

Department of Electrical and Electronic Engineering
Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong, China.
Email: {eejtzhou, zhiqin, eecyan, eeau}@ust.hk

ABSTRACT

This paper addresses the security issues of the multimedia encryption schemes based on multiple Huffman table (MHT). A detailed analysis of known-plaintext attack is presented to show that the Huffman tables used for encryption should be carefully selected to avoid the weak keys problem. Further, we propose an efficient chosen-plaintext attack on the basic MHT method as well as the enhanced scheme inserting random bits. We also show that random rotation in partitioned bit stream cannot essentially improve the security.

1. INTRODUCTION

The development in digital multimedia and communication technologies has paved ways for people around the world to acquire, utilize, and share multimedia content. The multimedia data security, as a consequence, is becoming increasingly important.

The intuitive solution is to encrypt the multimedia data using conventional cryptographic algorithms such as DES. Due to the high data rate of multimedia data, these usually add large number of processing overhead to meet the real-time delivering requirement. To reduce the computations involved, selective encryption has been proposed. However, portions of the content are still visible and identifying the features introduces some additional computations. Therefore, designing a good multimedia encryption scheme, which features high level of security and low computational cost, is a challenging task.

Recently, Wu *et al.* proposed an efficient multimedia encryption methodology, called multiple Huffman table (MHT) method [1-3], which combines the encryption with entropy coding using multiple statistical models alternatively in a secret order. The major advantage of this kind of scheme is that the encryption with reasonably high level of security and unaffected compression can be achieved simultaneously, requiring almost negligible additional overhead. Unfortunately, MHT method is only secure under cipher-only and know-plaintext attack, and is vulnerable under chosen- plaintext attack [1-4]. To improve the security, several kinds of enhanced MHT encryption schemes have been proposed by either inserting random bits in the generated cipher or introducing stream cipher into the original systems [1-4]. Recently, a relevant encryption scheme via random rotation in partitioned bit streams has been reported, and has been integrated with MHT [5-6].

In this work, we analyze the security of the basic MHT scheme as well as the enhanced methods. We show that the Huffman tables used for encryption should be carefully selected, otherwise the computational cost needed by exhaustive search will be significantly reduced. An efficient chosen-plaintext attack on the enhanced method inserting random bits is also presented. Furthermore, we briefly evaluate the security of the scheme with random rotation in partitioned bit stream by using a simple example.

The rest of the paper is organized as follows. Section 2 presents an overview of MHT methods. The detailed security analysis is given in section 3. Section 4 makes some concluding remarks.

2. MHT ENCRYPTION METHOD

Encryption and compression are, in fact, intimately related in that it is redundancy in the data permits encryption and compression [7]. This fact motivates us to integrate encryption with compression.

Huffman coder is very popular in modern multimedia compression system, with the aim of compressing symbols such as quantized DCT coefficients into bit streams, according to some predefined statistical models. Since, usually, the statistical model space is not large, it does not offer enough security. The MHT algorithm [1-4], in order to increase the model space while maintaining the computational efficiency, keeps the structure of the Huffman tree but enlarges the model space through mutating the original trees. The procedure of basic MHT algorithm can be described as follows:

a) Train four original Huffman trees from different sets of training data. An example for JPEG dc coefficient coding can be found in Fig. 8 in [1].
b) Perform tree mutation to create the whole Huffman tree space. The operation is illustrated in Fig. 1.
c) Randomly select m different trees from the space, and number them from 0 to $m-1$.
d) Generate a random vector $P=\{p_0,\cdots,p_{n-1}\}$, where each p_i is an integer ranging from 0 to $m-1$.
e) For the *i*th symbol, use tree $p_{i \bmod n}$ to encode it.

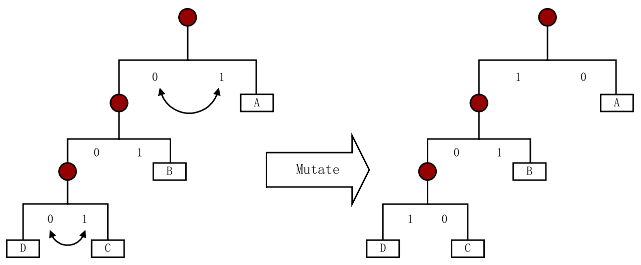

Fig. 1. Tree Mutation Process. A, B, C, D denote alphabets.

Typically, we set $m=8$ and $n=128$. Since both the creation of Huffman trees and the look-up table operations are quite simple, the MHT scheme needs small key-setup cost and encryption/decryption cost. On the other hand, due to the fact that decoding a Huffman coded bit stream without any knowledge of the Huffman table is extremely difficult [8], MHT is secure under cipher-only attack and know-plaintext attack [1-3]. However, under chosen-plaintext attack the MHT scheme is vulnerable, even when the cipher can receive symbols as a whole chunk and output the corresponding codewords all together [1].

Aiming to increase the security against chosen-plaintext attack, two types of enhanced schemes: selective random bit insertion scheme, and stream cipher involved MHT scheme, have been proposed [1,4]. In the former case, another random vector $Q=(q_0,\cdots q_{v-1})$ is generated [1]. For the $(w\times i)$ th bit in the encrypted bit stream, where $w\geq 50$ is a constant and $i\in\mathbb{N}$, function F_i will be performed.

$$F_i(c)=\begin{cases} \textit{do nothing} & \textit{when } q_{i \bmod v}=0 \\ \textit{add one random bit after } c & \textit{when } q_{i \bmod v}=1 \end{cases} \quad (1)$$

In the latter case, a key stream $h(s_0)$ is generated using a stream cipher such as SHA-1, where s_0 is the seed [4]. The key stream will be partitioned into several blocks $b_1 \| b_2 \| \cdots \| b_r \|$, where b_i determines a Huffman tree from the tree space to encode the encountered symbol. See [1,4] for detailed discussion.

Recently, another relevant encryption scheme via random rotation in partitioned bit streams was proposed by altering the generated bit stream [5]. Suppose $X=x_1x_2\cdots x_N$ is a bit stream of N bits. This algorithm consists of the following two major steps, where p_i and r_i serve as the secret keys.

1) Partition X into k blocks X_i with length p_i, $i=1,\cdots k$.
2) Perform a r_i-bit left rotation on each block A_i.

An integrated random rotation scheme with stream cipher has also been reported in [6].

3. SECURITY ANALYSIS

3.1 Known-plaintext attack and weak keys problem

In this subsection, we propose a detailed analysis of known-plaintext attack on basic MHT, and show its weak keys problem. Suppose we are given a bit stream $X=x_1x_2\cdots x_N$, and know that it is the multiple Huffman encoding of some alphabet string $C=c_1c_2\cdots c_S$. Since any bit stream is of equal probability, we have no knowledge to determine the boundary of X, i.e., establish the alphabet-codeword relationship. Hence, in order to partition X into S nonempty sub-blocks $d_1d_2\cdots d_S$, we have to consider C_S^{N-1} cases. In the traditional case where just one single Huffman table is used, we can check the prefix condition and consistence condition to eliminate some impossible partitions. In the current MHT scheme, we cannot do so since these two conditions are not necessarily satisfied for the whole bit stream. However, if we can estimate the upper bound K to the length of a codeword, the number of partition can be reduced from $C_S^{N-1}=O(N^S)$ to $O(K^S)$. According to the Huffman tree creation procedure and Fig. 8 in [1], we know $K=11$. Nonetheless, an attack with complexity of $O(11^S)$ is still formidable.

A more sophisticated method would be to use the fact that we have knowledge about the tree structure, i.e., we know the possible length of any alphabet, as Table 1 shows. In the limiting case where all the m trees are mutated from the same original tree, identical alphabets in the plaintext correspond to codewords with the same length, although they might be encoded by different trees. And from Table 1, we can find that one specific alphabet can take at most 3 different lengths. This observation immediately reduces the number of cases to be checked to $3^{|\Omega|}=3^{13}$, since we only need to guess the length once for all identical alphabets. Here, Ω denotes the alphabet set $\{a_0,a_1,\cdots,a_{11},error\}$. The complexity of order $3^{|\Omega|}$ is already feasible, especially when the given plaintext dose not include all $|\Omega|$ different letters.

In both basic and enhanced MHT, the Huffman trees are randomly selected from the tree space. The probability that all selected trees are mutated from the same original tree is:

$$P_1=4\prod_{i=0}^{m-1}(M-i)/(4M-i)\approx 6\times 10^{-5} \quad (2)$$

where $M=2^{12}$ is the total number of trees generated from one original tree. It is almost four times higher than that of the optimal case where all m selected trees are uniformly mutated from the four original trees

$$P_2 = M^4(M-1)^4 / \prod_{i=0}^{m-1}(4M-i) \approx 1.52\times 10^{-5} \qquad (3)$$

Note, also, that if all the m trees come from the last two original trees, around $2/3$ of all alphabets in Ω will have fixed length. The occurring probability is:

$$P_3 = \prod_{i=0}^{m-1}\frac{2M-i}{4M-i} \approx 0.0039 \qquad (4)$$

Hence, random selection of Huffman trees will introduce weak keys problem in the sense that some of the selected trees provide lower security than the others. To avoid this problem, a better solution is to randomly select $m/4$ trees among each original tree space of size M, instead of choosing m trees from the whole space of size $4M$.

Table 1: Possible code length for alphabets. In 'length' column, four numbers correspond to four original trees.

Alphabet	Length	Alphabet	Length
a_0	2 2 2 1	a_7	5 7 6 6
a_1	3 2 2 3	a_8	6 8 7 7
a_2	3 2 3 4	a_9	7 9 8 8
a_3	3 3 3 4	a_{10}	8 10 9 9
a_4	3 4 3 3	a_{11}	9 11 10 10
a_5	3 5 4 4	error	9 11 10 10
a_6	4 6 5 5	-	-

3.2 Cryptanalysis on the scheme inserting random bits

Before presenting the cryptanalysis on the enhanced scheme inserting random bits, we propose an efficient chosen-plaintext attack on the basic MHT method.

Recall in [4], the author proved, under chosen-plaintext attack, the basic MHT could be broken in $n|\Omega|$ times of encryption oracle access. But this conclusion is only valid when the cipher receives *one single* symbol and outputs its corresponding codeword. In this work, we consider the more complicated case where a whole chunk of alphabets are received and the corresponding codewords are output all together. We have the following proposition:

Proposition: Under chosen-plaintext attack, the basic MHT can be broken in $n^2|\Omega|$ oracle accesses, when the cipher outputs the bit stream of n alphabets all together.

Proof: The attack method can be described as follows.

Step 1: Input alphabet stream $c_1c_2\cdots c_{n-1}a_0$, and obtain bit stream $z_1z_2\cdots z_{n-1}z_n$, where $c_i \in \Omega$ can be any alphabet, a_i is *i*th alphabet, and z_i is the corresponding codeword.

Step 2: Input another length-*n* alphabet stream $c_1c_2\cdots c_{n-1}a_{11}$, and the output sequence is $z_1z_2\cdots z_{n-1}z_n^{'}$. We choose a_{11} is because, from Fig. 8 in [1], a_0 and a_{11} are separated by the root node, i.e., the first bit of the codeword representing a_0 and a_{11} is different. Therefore, we can obtain $Z^{'} = z_1z_2\cdots z_{n-1}z_n00\cdots 0 \oplus z_1z_2\cdots z_{n-1}z_n^{'}$. Denote the index of the first '1' in $Z^{'}$ as j (left to right order). Obviously, the codeword of a_0 and a_{11} starts from the *j*th bit.

Step 3: Input length-*n* alphabet stream $c_1c_2\cdots c_{n-1}a_1$. Since the start position has been determined in *Step 2*, it is straightforward to find the corresponding codeword of a_1.

Continuing in this fashion, in *Step* $|\Omega|$, all the alphabet-codeword pairs of the *n*th tree are recovered. Repeat such steps, we can restore all the alphabet-codeword pairs for all n trees, and the computational cost is $n \bullet n|\Omega| = n^2|\Omega|$. □

Now the key thing for the random bits insertion scheme is to find the locations of the inserted bits. Note that these locations have been fixed as long as the vector Q is determined. Assume the encoded bit stream of $c_1c_2\cdots c_n$ is $B = b_1b_2\cdots b_f y_1 b_{f+1}\cdots b_g y_2 b_{g+1}\cdots b_h y_3 b_{h+1}\cdots$, where b_i is the original bit, and y_i is the inserted random bit taking 0 or 1 with the same probability. Therefore, we can encode $c_1c_2\cdots c_n$ T times, and obtain $B_1, B_2\cdots B_T$. We then perform the operations $D_i = B_1 \oplus B_i$, $i = 2 \sim T$. Due to the randomness of y_i, the bit '1' in D_i reveals where the inserted bits locate. The missing detection probability for every random bit is 2^{-T}. After successful detection, we simply remove them from the cipher, and apply the attack described in the proof to break the enhanced scheme.

Since when detecting the random bits, we exploit the random property, one may argue that what if inserting deterministic bits instead of random bits. In fact, this does not help much to improve the security. Recall that the random bits will be added after the $(w\times i)$ th bit if $q_{i \bmod v} = 1$. Hence, in the bit stream available, the $(w+1)$ th bit is a suspicious bit to be the first added bit, and $(2w+1)$ th, $(2w+2)$ th bit are two suspicious bits to be the second added bit. Along this line, we can find all suspicious bits. And according to [1], the number of random bits should not exceed 1% of the original ciphertext, thus, the total number of suspicious bits is quite limited. For example, for a cipher with length 700, and $w = 70$, the total number of suspicious bits is 45. Consequently, after recovering all the alphabet-codeword pairs for each tree, using the aforementioned method, we further perform two additional checking steps:

Step a: Check whether the obtained codewords contain suspicious bits according to the suspicious bits index. If yes, go to *Step b*, otherwise, continue to find the alphabet-codeword pairs for the previous tables.

Step b: Check whether the following equality holds, where l_i denotes the length of obtained codeword of a_i.

$$E = \sum_{i=0}^{|\Omega|-1} 2^{-l_i} = 1 \qquad (5)$$

Since Huffman coding is optimal, Eq. 5 will be violated if any bits are inserted. In the case of $E < 1$, from Table 1, we can find the relationship between the inserted bit location $u \in N$ (relative position within the codeword, left to right order) with $Y = -\log_2(1-E)$, for four original tree conditions. From Fig. 2, we can see that if $u \ge 6$, we can obtain an unique solution of u from Y. When $2 \le u < 6$, the number of solution is at most 2. Under this situation, we just keep these two candidates, and one of them can be used to successfully remove the inserted bit, leading to correct decoding of the bit stream.

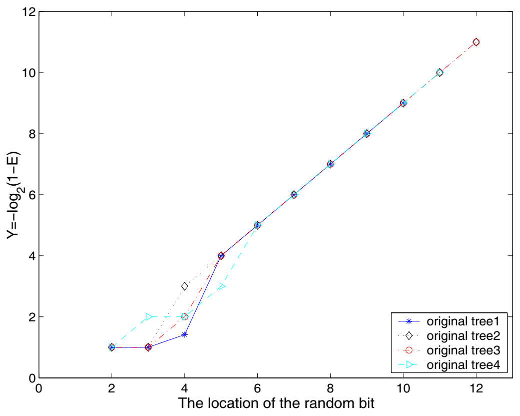

Fig. 2. Relationship between inserted bit location u and Y.

3.3 Random rotation in partitioned bit streams

A relevant encryption scheme via random rotation in partitioned bit streams has been suggested in [5]. Due to the space limitation, we just present a simple example to illustrate the basic idea of our attack, and the detailed analysis will be reported in our forthcoming paper. Suppose there is one alphabet a corresponds to codeword '1', and another alphabet b corresponds to codeword '01'. Note that in this encryption scheme, multiple Huffman tables are not considered. We set the partition key $p = 6$, and rotation key $r = 3$. Our goal is to recover these two values.

When encoding length-*n* alphabets $aa\cdots a$, we can obtain bit stream $11\cdots 1$. We then encode $baa\cdots a$, and the obtained bit stream is $111011\cdots 1$. From this, we can know that $p - r = 3$ holds. In the following steps, we can input $aba\cdots a$, $aaba\cdots a$, $aaaba\cdots a$, and get the corresponding bit stream $111101\cdots 1$, $1111101\cdots 1$, and $011\cdots 1$. It can be seen that '0' in the codeword of $aaba\cdots a$ hits the boundary, i.e., $p = 6$. So, we can derive $r = 3$.

Although the author has proved the existence of large numbers of alias keys that encode the same plaintext to the same ciphertext [5], the underlying reason of the success of the attack is that by differential analysis, the number of alias keys can be reduced to 1. The analysis for MHT with random rotation is a natural extension of the above method. Therefore, random rotation in partitioned bit streams cannot essentially improve the security.

4. CONCLUDING REMARKS

The security of the multimedia encryption schemes based on multiple Huffman table has been addressed. A detailed analysis of know-plaintext attack reveals that the multiple Huffman tables used for encryption should be carefully selected to avoid the weak keys problem. In some extreme cases, the computation of exhaustive search can even be of order 3^{13}. Furthermore, we propose an efficient chosen-plaintext attack to the basic MHT scheme as well as the enhanced scheme inserting random bits. We also briefly evaluate the security of a relevant encryption scheme with random rotation in partitioned bit streams.

In this paper, we have not discussed the schemes with stream cipher generators. Generally speaking, the only relevant attack model in those cases is the known-plaintext attack because the internal state of the stream cipher is independent of the plaintext and the ciphertext, making the flexibility of chosen-plaintext attack cannot be exploited. Currently, it is not practical to recover the secret keys of the stream cipher such as SHA-1, although recently breakthrough has been achieved [9]. The security of these schemes, however, solely relies on the assumption that decoding a multiple Huffman table encoded bit stream is computationally infeasible even if we have full knowledge about all the multiple tables, but only do not know the using order. To what extend is this assumption true is still an open question, and needs more mathematical analyses.

5. ACOKNOWLEDGEMENT

This work has been supported by the Innovation and Technology Commission of the Hong Kong Special Administrative Region, China (project no. ITS/122/03 and project no. GHP/033/05).

6. REFERENCES

[1] C. Wu and C.-C. J. Kuo, "Design of integrated multimedia compression and encryption systems", *IEEE Trans. Multimedia,* vol. 7, no.5, pp. 828-839, 2005.
[2] C. Wu and C.-C. J. Kuo, "Efficient multimedia encryption via entropy codec design", *SPIE international symposium on electronic imaging,* San Jose, CA, Jan 2001.
[3] C. Wu and C.-C. J. Kuo, "Fast encryption methods for audiovisual data confidentiality ", *SPIE international symposium on Information Technologies,* Boston. pp. 284-295, Nov 2000.
[4] D. Xie and C.-C. J. Kuo, "Enhanced multiple Huffman table (MHT) encryption scheme using key hoping", *Proc. ISCAS 2004*, vol.5, pp. 568-571, May 2004.
[5] D. Xie and C.-C. J. Kuo, "Multimedia data encryption via random rotation in partitioned bit streams", *Proc. ISCAS 2005* vol.5, pp. 5533-5536, May 2005.
[6] I. Cheong, Y. Huang, Y. Yung, S. Ke, and W. Chen, "An efficient encryption scheme for MPEG video", *International Conference on Consumer Electronics,* pp. 61-62, Jan 2005.
[7] S. T. Klein, A. Bookstein, and S. Deerwester, "Storing test retrieval systems on CD-ROM: Compression and Encryption", *ACM Trans. Information Systems,* vol. 7, no.3, pp. 230-245, 1989.
[8] D. Gillman, M. Mohtashemi, and R. Rivest, "On breaking a Huffman code", *IEEE Trans. Information Theory,* vol. 42, no. 3, pp. 972-976, 1996.
[9] X. Wang, Y. Yin, and H. Yu, "Finding collisions in the full SHA-1", *Lecture notes on computer science,* vol. 3621, pp. 17-36, , Springer-Verlag, Berlin, 2005.