

Efficient Multimedia Encryption via Entropy Codec Design

Chung-Ping Wu and C.-C. Jay Kuo
Department of Electrical Engineering-Systems
University of Southern California, Los Angeles, CA 90089-2564
E-mail: {chungpin, cckuo}@sipi.usc.edu

ABSTRACT

Efficient encryption algorithms are essential to multimedia data security, since the data size is large and real-time processing is often required. After discussing limitations of previous work on multimedia encryption, we propose a novel methodology for confidentiality, which turns entropy coders into encryption ciphers by using multiple statistical models. The choice of statistical models and the order in which they are applied are kept secret as the key. Two encryption schemes are constructed by applying this methodology to the Huffman coder and the QM coder. It is shown that security is achieved without sacrificing the compression performance and the computational speed. The schemes can be applied to most modern compression systems such as MPEG audio, MPEG video and JPEG/JPEG2000 image compression.

Keywords: multimedia encryption, confidentiality, Huffman coding, QM coder, selective encryption, signal scrambling

1. INTRODUCTION

The secrecy of transmitted or stored data is required in a variety of applications, and multimedia data is no exception. Since the emerging Internet and wireless network are open networks which are vulnerable to eavesdropping, confidentiality protection is especially important for reliable network applications. Internet telephony, Internet conferencing, Internet security camera and multimedia databases are a few examples of promising network multimedia applications that require confidentiality.

Over the past few decades, the cryptographic society has provided numerous successful confidentiality protection schemes, which are mainly focused on the security of alphanumeric data. Although encrypting the whole audiovisual data stream with cryptographic ciphers achieves security, the large size of audiovisual data requires a considerable amount of computation power. Since audiovisual data usually contain much lower information value density (information value per data size) than text data, enciphering schemes with significantly lower computational cost is desired. Moreover, encryption/decryption speed is especially critical to multimedia data because real-time processing is often required.

Some may argue that today's desktop computers are fast enough to perform both multimedia compression and total encryption together in real-time, which means the total encryption method is good enough for desktop multimedia applications. While this may be true, there are emerging applications in mobile computing and server-end computing. The computing speed of palm-top (or even wrist-watch) computing devices may not be fast enough for total encryption. For server-end computing, a server has to handle hundreds of processes at the same time. The reduction in the computational cost of each process would allow more processes to be executed. In fact, there are special servers¹ designed to handle encryption and decryption, so that the main server could be freed from these computationally extensive operations. We may eliminate the need for a dedicated encryption server by using low computation cost audiovisual encryption algorithms.

Traditional solutions to audiovisual data confidentiality are based on scrambling techniques, which consist of relatively simple permutation and/or affine transformation operations in the time or the frequency domain. As the computing power increases quickly these days, scrambling algorithms become vulnerable to attacks. More recently, the focus of multimedia encryption research has been shifted to selective encryption, where some part of the audiovisual bitstream is enciphered with cryptographical encryption. However, selective encryption is limited in its range of applications (see discussion in Section 2). Therefore, we propose a novel multimedia encryption methodology that turns the entropy coder into ciphers by using multiple statistical models alternatively in a secret order.

The encryption cipher and the entropy coder do bear some resemblance. Both of them turn the original data into redundancy-free bit streams, which cannot be decoded without certain information. The information is the key and the statistical model for encryption and entropy coding, respectively. It is interesting to explore the feasibility of hiding the statistical model information to completely prevent decoding of the compressed bit stream. Simple entropy codecs such as the Huffman and the QM coders are very popular in modern multimedia compression standards, yet they do not offer a model space that is large enough for security. In order to increase the model space while maintaining the low computational cost of these coders, we keep their basic structures unchanged but enlarge the statistical models used for symbol coding.

This paper is organized as follows. Previous work on multimedia encryption is examined in Section 2. The methodology of encryption via multiple statistical models in entropy coding is discussed in Section 3. A fast encryption scheme by using multiple Huffman coding tables is proposed in Section 4. A novel encryption scheme based on the use of multiple state indices in the QM coder is presented in Section 5. Finally, concluding remarks are provided in Section 6.

2. PREVIOUS WORK ON MULTIMEDIA ENCRYPTION

In this section, we review two main approaches to achieve multimedia encryption, and explain the reason why they may not be compatible with requirements of modern multimedia compression systems.

2.1. Signal Scrambling

Early work² on signal scrambling used analog devices to permute the signal in the time domain or distort the signal in the frequency domain by using filter banks or frequency inverters. These schemes have high residual intelligibility,³ and are relatively easy to crack using modern computers.⁴ As digital signal processing became popular, the focus was shifted to scrambling in the domain of orthogonal transforms, such as DFT,^{5,4} DCT,^{6,3} the wavelet transform⁷ and the Hadamard transform.⁸ These new transform domain scrambling techniques have much lower residual intelligibility than old ones. However, they are still much more vulnerable to cryptanalysis than encryption. Since scrambling techniques usually involve only permutation or affine transformations, they are very weak to known-plaintext³ and chosen-plaintext attacks. Some successful plaintext-only attacks have also been reported.^{9,3}

Furthermore, scrambling of the raw signal degrades the performance of multimedia compression systems, which have been designed base on the characteristics of unscrambled signals. While this problem could be alleviated by carefully designing the scrambling technique,^{5,10} deterioration in compression efficiency is inevitable. Since almost all multimedia data are compressed before transmission today, confidentiality protection schemes that are completely compatible with multimedia compression are more favorable.

2.2. Selective Encryption

The basic concept of selective encryption is to select the most important coefficients from either final results or intermediate steps of a compression system, and then encrypt them with conventional cryptographic ciphers such as DES. The coefficients *not* selected are sent to the transmission channel (or storage media) with no encryption at all or only with light-weight scrambling. In order to be effective, the selected portion must be closely tied with the intelligibility of the particular host media, and its size must be significantly smaller than that of the whole data stream. Whether a selective encryption method can be successfully built for a media compression system depends on the degree of concentration of intelligibility in the compressed data.

Most existing selective encryption schemes are aimed at either JPEG or MPEG, and can be naturally extended to the other because both of them are based on the 8×8 block DCT followed by scalar quantization, run-length coding and Huffman coding. We can almost say that MPEG video compression uses JPEG to compress its I-frames. Early work in the field proposed to encrypt all I-frames in the MPEG video stream.^{11,12} It is argued that B frames and P frames cannot be reconstructed without I frames. However, Agi and Gong¹³ showed that significant portions of the video are still visible under this scheme due to the unencrypted I-macroblocks in the B and P frames. These I-macroblocks are fully decodable without any information from the I-frames. Thus, they proposed to encrypt all I-frames and all I-macroblocks in B and P frames. The main problem with this approach is that I-frames alone already occupy about 30% \sim 60%¹³ or more¹⁴ of the MPEG video stream. If the I-macroblocks are added, the percentage is even higher. Consequently, the cost of selectively encrypting all I-frames and I-macroblocks is close to

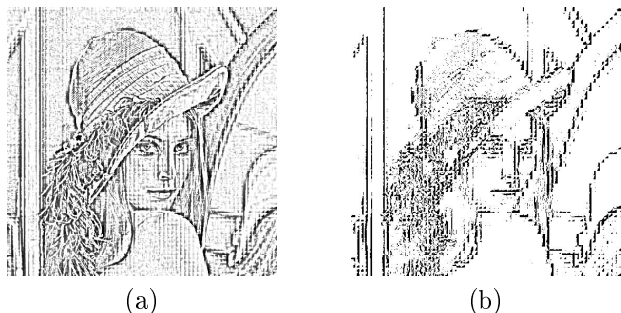


Figure 1. (a) The effect of replacing DC coefficients with a constant value where all DC coefficients are assigned to 128, and (b) the effect of discarding sign bits in every DCT coefficient where all AC coefficients take positive values. In both figures, images are enhanced by operations such as brightness modification, sharpening or thresholding.

that of total encryption. Moreover, identifying I-macroblocks in the MPEG stream introduces some computational overhead,¹⁴ which further decreases the gain of selective encryption.

Since both the I-frames and the I-macroblocks are entirely composed of DCT coefficients, researchers tried encrypting only some of DCT coefficients so that computational speed could be further increased. It is a well-known fact that in the DCT domain, most of the image energy is concentrated in the DC coefficient. In most images, the DC coefficient alone contains more energy than all AC coefficients combined. Based on this knowledge, Tang¹⁵ proposed a system, which encrypts DC coefficients with DES and scrambles AC coefficients with block-based permutation. However, *energy concentration* is often unrelated to *intelligibility concentration*. Figure 1(a) shows a Lena image reconstructed after discarding all information of DC coefficients (i.e. all DC coefficients are set to 128). One can see that the semantic content of the image is almost unaffected. Therefore, the security level of Tang's system is reduced to that of the block-based permutation, which is vulnerable to known-plaintext¹⁶ and chosen-plaintext attack. Since different AC coefficients possess different statistical characteristics, it was proven¹⁷ that low-resolution images could be reconstructed even by using ciphertext alone. It was shown in our previous work¹⁸ that encrypting more low-frequency AC coefficients cannot fully destroy the image content either.

Since selectively encrypting *some* DCT coefficients does not work, one may try to encrypt some part of *every* DCT coefficient. Shi and Bhargava¹⁹ proposed to encrypt the sign bit of every DCT coefficient in JPEG compression (and every motion vector in MPEG²⁰). It was further shown¹⁸ that useful image content could be recovered from the output of this scheme, and encrypting more significant bits in every DCT coefficient does not work either.

Another problem with algorithms discussed above is that they decrease compression efficiency as a result of encrypting DCT coefficients before run-length and Huffman coding. Qiao and Nahrstedt¹⁴ proposed an MPEG video encryption system, which performs encryption *after* the Huffman coding stage so that compression ratio is unaffected. It is based on encrypting *every other bit* (one bit in every two bits) in the MPEG bit stream with DES, and they provided convincing reasoning for the security of this scheme. However, since one half of the bitstream has to be encrypted, the computation needed is at least 50% of total encryption, not including the overhead introduced by the selection process. Since this selection process ignores the issue of which bits are more important to intelligibility, it unlikely to be a good selection policy.

In summary, selective encryption is conceptually straightforward, but not compatible with multimedia compression systems possessing any of the following two characteristics.

1. Based on an orthogonal transform followed by quantization

Although the signal energy is usually concentrated to a few coefficients after orthogonal transforms, such is often not the case for intelligibility. We have observed that intelligibility tends to be scattered among all frequency components. This property was clearly explained earlier in encrypting DCT coefficients of JPEG/MPEG compression. Similar results have also been observed in MPEG audio compression. In contrast, model-based media compression systems are more suitable for selective compression because they are likely to contain key-coefficients that control intelligibility.

2. Containing entropy coding at the last stage

When selective encryption is performed before the entropy coding stage, the compression performance of entropy coding is decreased. As a result, some researchers concluded that "*there is a contradiction between compression and encryption*".⁵ Similar statements can be found in.^{15,10} If we perform selective encryption after entropy coding, it is difficult to distinguish which bits are more important to intelligibility.

Popular multimedia compression systems such as JPEG/JPEG2000 image compression, MPEG video compression and MP3 audio compression have both characteristics described above. They are transform-based followed by quantization and entropy coding. Thus, it is difficult to find an effective selective encryption scheme for them. In the next section, we propose a fast encryption methodology for systems that are not suitable for selective encryption.

3. ENCRYPTION BY USING MULTIPLE STATISTICAL MODELS IN ENTROPY CODERS

Since the entropy coder at the last stage of a media compression system prohibits the successful use of selective encryption, the feasibility of integrating encryption into entropy coding is investigated in this section. We will first review previous work on integrating encryption with entropy coding, and then propose a method for turning simple entropy coders into encryption ciphers.

Most previous work in the area of integrating encryption into entropy coding has been focused on "adaptive arithmetic coding using higher-order models".²¹⁻²⁵ This compression algorithm is the state-of-the-art system for text data compression. It was converted into a cipher by hiding the initial statistical model. This entropy coder is favored because its statistical model changes constantly and the model space is enormous, which prohibits brute-force exhaustive searching. Since there is essentially no additional computation needed to achieve encryption, this scheme is faster than any other encryption system. It is generally believed that this scheme is very secure against ciphertext-only and known-plaintext attacks. Successful chosen-plaintext attacks were discovered (e.g. as described in work²¹⁻²⁴), but in applications where the chosen-plaintext attack is not a threat, this scheme is very efficient in performing text compression and encryption at the same time.

The Huffman coder and the QM coder are the two most popular entropy coders in multimedia compression systems, and both have very simple statistical models. The statistical model of the Huffman coder is often a fixed-size non-adaptive binary tree, and the initial state of the QM coder consists of only 3 integer numbers (A, C and I).²⁶ It has been shown¹⁹ that hiding the Huffman coding table is not an ideal approach for encryption. Hiding the initial state of the QM coder also provides no security because it is easy to try all possible values of these integers.

In order to overcome the problem of a limited key/model space in a simple entropy coder, we propose to use m statistical models instead of only one. The m models are used alternatively to encode the input symbol stream. When these models are swapped in a fixed and known fashion, the size of the overall model space only increases linearly with m . However, if the models alternate in a hidden order, the size of key space increases with m^p , where p is the length of the hidden alternating sequence. m should be kept quite small because it is linearly related to the memory required to store these statistical models. Setting m to 4 and p to 64 generally produces a large enough key space for security.

In the next two sections, two encryption schemes are presented as examples to demonstrate how the multiple statistical model methodology works. One example is encryption by using multiple Huffman coding tables, and the other is encryption by using multiple indices in the QM-coder estimation state machine.

4. MULTIPLE HUFFMAN CODING TABLES ENCRYPTION SCHEME

Huffman coding with a predefined fixed Huffman table is used in most modern multimedia compression systems, including MPEG video, MPEG audio and JPEG image compression. Since coding of each symbol is only a simple table lookup process, it is without doubt the fastest compression algorithm. In this section, we first present and analyze the basic (core) algorithm of the multiple Huffman tables scheme (hereafter denoted as the MHT-encryption scheme). Then, two ways of enhancing the security of the MHT-encryption scheme are described.

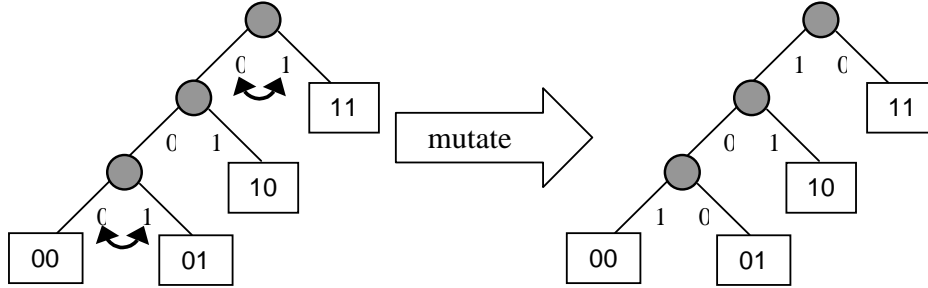


Figure 2. Illustration of the Huffman tree mutation process.

4.1. Basic Algorithm for the MHT-encryption Scheme

The input data stream is encoded using multiple Huffman coding tables. The content of these tables and the order that they are used are kept secret as the key for decryption. The basic algorithm can be described in the following 3 steps.

1. Generate 2^k different Huffman coding tables. They are numbered from 0 to $(2^k - 1)$.
2. Generate a random vector $P = (p_1, \dots, p_n)$, where each p_i is a k -bit integer varying from 0 to $(2^k - 1)$.
3. For the i_{th} symbol in the original data stream, use table $p_{(i-1 \pmod n)+1}$ to encode it.

It is important to generate the Huffman coding tables such that the compression ratio will not be affected. One way to achieve this goal is to generate each Huffman table from a different set of training images (or audio pieces). Although the resulting Huffman tables are different from each other, every Huffman table is equally “good” as long as each training set is a balanced representation of all audio pieces (or images) in the world. If two Huffman tables happen to be identical, we can simply pick up one of them. By using this approach, thousands of different Huffman tables can be generated and published. In Step 1 of the algorithm, we may randomly choose 2^k tables from the space of all available tables and send the indices as the key. If there is a total of 2^N published Huffman coding tables, each index would be N -bit long.

In the proposed system, we adopt another easier method to generate Huffman coding tables. Instead of training thousands of Huffman coding tables, we only train and publish 4 different Huffman tables, and millions of different tables could be derived by using a technique called Huffman tree mutation. As shown in Figure 2 (a), a standard Huffman tree has every left-hand-side branch labeled ‘0’ and every right-hand-side branch labeled ‘1’. If we permute the label-pairs as indicated in Figure 2, we will get a new Huffman tree as shown in Figure 2 (b). We call it the *Huffman tree mutation* process. Please note that each label-pair is attached to one inner node. For a Huffman table with m entries, its Huffman tree would have m leaves and $(m - 1)$ inner nodes and label-pairs, which provides us the opportunity to make $(m - 1)$ decisions about whether to permute each label-pair. Therefore, 2^{m-1} Huffman trees can be derived. In order to produce a new Huffman tree, we first randomly generate an $(m - 1)$ bit integer, then permute the label-pairs in the standard Huffman tree if the corresponding bit in the integer is zero. It should be noted that Huffman tree mutation has absolutely no influence on the coding efficiency of a Huffman tree.

The MHT algorithm can be illustrated with the following example. We choose the Huffman coder for JPEG DC coefficients because it has the right size for the demonstration purpose. Figure 3(a) shows the original Huffman tree used in encoding JPEG DC coefficients. It is created by using the statistics collected from a set of images that is supposed to properly represent all of the images that the JPEG system will encode. Figure 3(b)~(d) are three other Huffman trees trained with three independent image sets. The four Huffman trees in Figure 3 are used as basic trees to derive others through Huffman tree mutation. Since each of these basic trees can be mutated into 2^{13-1} different Huffman trees, we will have a total of $4 \times 2^{13-1} = 2^{14}$ trees. Next, these trees are used in the MHT algorithm as described above. k and n are set to 14 and 64, respectively. As shown in Table 1, several images from the JPEG2000 evaluation image set are used to compare the compression performance of MHT with that of the original Huffman tree. For some images, the original Huffman tree provides a smaller compressed data size. However, for some others,

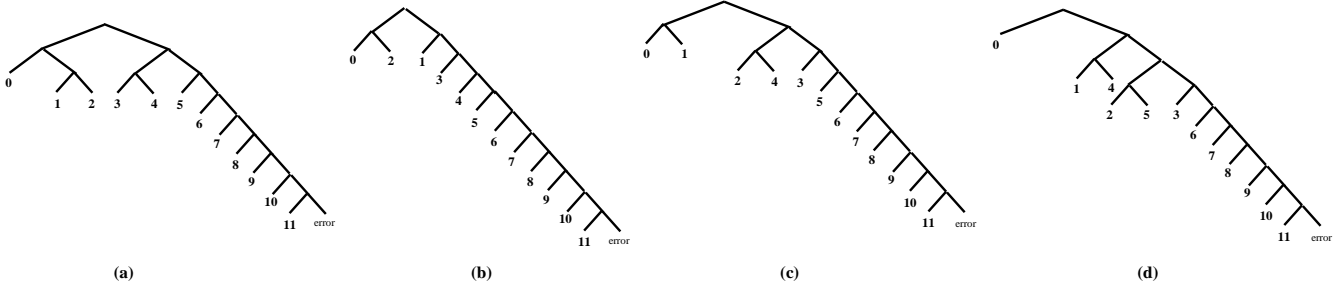


Figure 3. The original Huffman coding tree for JPEG DC coefficient coding is given in (a) while three Huffman trees trained from three image sets are shown in (b), (c) and (d).

Image name	Total size (bits) of encoded DC coefficients with the standard Huffman tree	Total size of encoded DC coefficients with the MHT algorithm	Percentage of size increase
bike	248619	262062	5.41
cats	242811	238112	-1.94
chart	176158	183949	4.42
cmpnd1	15233	15451	1.43
cmpnd2	1526488	1598080	4.69
gold	19640	21147	7.67
hotel	19949	21727	8.91
mat	74819	73108	-2.29
seismic	11544	10942	-5.22
tools	86864	92396	6.37
water	118464	103983	-12.22
woman	233316	234119	0.34

Table 1. Compression performance comparison between the standard Huffman coding and the proposed MHT-scheme.

the MHT algorithm is better. The overall compression performance of MHT is approximately the same as that of the original Huffman coding.

Table 2 shows the codeword lengths of each symbol encoded with four basic Huffman trees. The Huffman trees mutated from a basic tree would have the same codeword lengths as the basic tree. As shown in Table 2, when all Huffman trees are used alternatively, there would be more than one possible codeword length for each symbol. For an attacker who does not know the order in which these trees are applied, synchronizing between the symbol stream and the encoded bit stream would be extremely difficult.

The net computational cost of the basic MHT-encryption scheme is less than one CPU operation per encrypted bit as explained below. When a symbol is to be encoded with a normal Huffman coder, the shift amount is added to the base address of the table to obtain the address of the desired Huffman code. This process is illustrated in Figure 4 (a). In the basic MHT system, we store the base addresses of the tables in a cyclic queue according to the order that they are used. When a symbol is to be encoded/encrypted, the base address is first loaded from the memory, and then the shift-amount is added to it. Afterwards, the index to the cyclic queue of base addresses should be increased by one. Then, the index should be compared with the end of the queue in order to decide whether it should be reset to the beginning of the queue. Therefore, the computational difference between our cipher/encoder and a normal Huffman coder is one memory-load, one addition and one comparison operation for each symbol encoded. The encoding process of the proposed cipher/encoder is shown in Figure 4 (b). Since each symbol in the original data usually have more than 3 bits in the Huffman bitstream, the net encryption cost of our algorithm is less than one CPU operation per encrypted bit.

Symbol name	Codeword length using Tree (a)	Codeword length using Tree (b)	Codeword length using Tree (c)	Codeword length using Tree (d)	Number of different codeword lengths
0	2	2	2	1	2
1	3	2	2	3	2
2	3	2	3	4	3
3	3	3	3	4	2
4	3	4	3	3	2
5	3	5	4	4	3
6	4	6	5	5	3
7	5	7	6	6	3
8	6	8	7	7	3
9	7	9	8	8	3
10	8	10	9	9	3
11	9	11	10	10	3
error	9	11	10	10	3

Table 2. The total number of different codeword lengths for each symbol when the four Huffman trees in Figure 3 are used together.

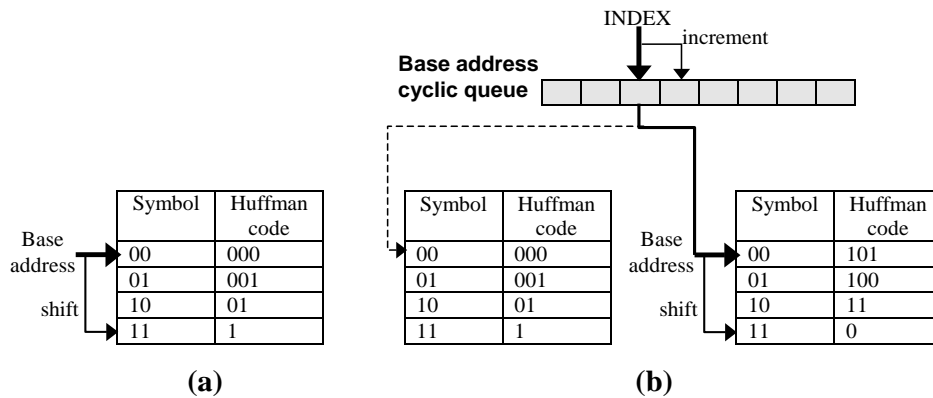


Figure 4. (a) The normal Huffman coder adds the shift amount to the base address of the table to obtain the address of the desired Huffman code, and (b) the proposed algorithm loads the base addresses of Huffman tables from a cyclic queue, and the index to the queue should be increased by one after coding of each symbol.

4.2. Security Enhancement for MHT-encryption

The security analysis of the basic MHT-encryption scheme is presented in our previous work¹⁸ and can be summarized as follows.

1. The proposed scheme is resistant to ciphertext-only exhaustive key search attack due to a very large key space.
2. Its resistance to known-plaintext attacks is based on the principle that synchronization between the plaintext and the ciphertext is extremely difficult to the attacker.
3. MHT-encryption is vulnerable to chosen-plaintext attack if the attacker could encrypt a large amount of chunks of chosen-plaintext and compare the resulting ciphertext.

Based on the cryptanalysis described above, we present two methods to enhance the security of the basic MHT-encryption scheme against known-plaintext and chosen-plaintext attacks. It is set as our design goal that the computational cost of the enhancement portion should be lower than that of the MHT-encryption scheme.

1. Selective random bit insertion in the encrypted bitstream.

As described earlier, MHT-encryption is secure against known-plaintext attack since it is difficult for the attacker to synchronize the plaintext with the ciphertext. In order to further increase the difficulty associated with synchronization, random bits can be inserted into the encrypted bitstream according to a secret key. The insertion algorithm is implemented as follows.

1. Generate a random vector $Q = (q_1, \dots, q_n)$, where each q_i is an 1-bit integer.
2. Perform function F_i on the $(w \times i)_{th}$ bit in the encrypted bitstream,

$$\begin{cases} F_i(B) = \text{do nothing}, & q_i \pmod n = 0, \\ F_i(B) = \text{add one random bit after } B, & q_i \pmod n = 1, \end{cases} \quad (1)$$

where w is a constant and $i \in N$.

This insertion method slightly increases the size of the ciphertext. The value of w should be larger than 50, so that the ciphertext size-increase would be less than 1%. The computational cost of this method is much less than one CPU operation per bit processed because bit-insertion is only performed once in every w bits.

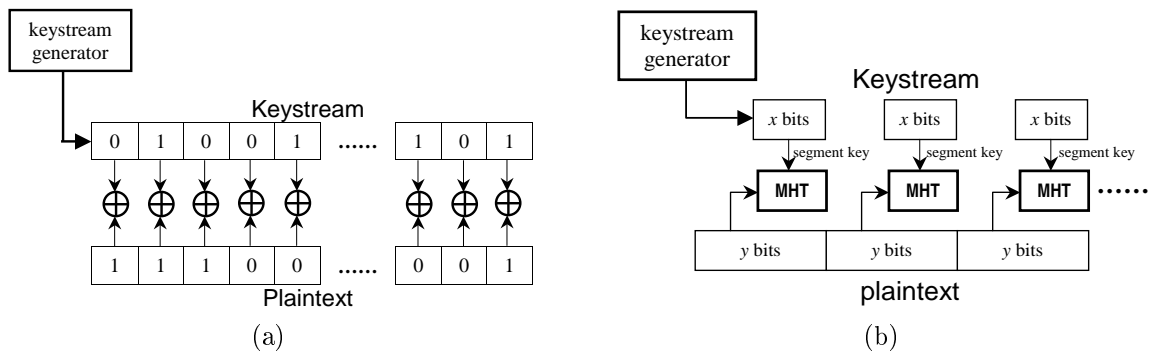


Figure 5. (a) The structure of a standard stream cipher and (b) an encryption system combining a stream cipher with the MHT-encryption algorithm.

2. Divide the plaintext into segments and use a different key for each segment

The stream cipher is integrated into our system under this method. A standard stream cipher uses a keystream generator to produce a pseudo-random binary sequence which has the same length as the plaintext. The ciphertext is generated by performing bit-wise XOR operation between the keystream and the plaintext. In order to ensure security, it is very important that the same keystream is never used more than once.²⁷ The main computational

load of stream ciphers is located in the keystream generator. It is very popular to use block ciphers such as DES to implement the keystream generator. Therefore, the computational cost of stream ciphers is approximately the same as that of block ciphers.

In order to greatly reduce the computation needed in the keystream generator, we divide the keystream and the plaintext into segments with a size of x bits and y bits, respectively. Instead of letting x equal to y as in normal stream ciphers, we make y at least 50 times larger than x . As shown in Figure 5(b), each segment of plaintext is enciphered using MHT-encryption, and the corresponding segment in the keystream is used as the *segment key*.

This scheme is mainly designed to solve the chosen-plaintext attack problem in MHT-encryption. As described earlier, the attacker could figure out the key for MHT-encryption by feeding many different chunks of attacker-selected data to the MHT-cipher. However, under this enhanced scheme, when enough plaintext is used to solve the segment key, the segment has ended and different segment keys are used in subsequent segments. Therefore, the segment keys obtained by the attacker cannot be used to decrypt the ciphertext to be encountered in the future.

Because x bits of keystream is used to encrypt y bits of plaintext, the computational cost of this enhancement method is

$$C_s \times \frac{x}{y}, \quad (2)$$

where C_s is the computational cost of a typical stream cipher. We recommend to set x and y to 320 and $16k$, respectively. Consequently, the cost of this method is 50 times smaller than the stream cipher, which is comparable to the cost of the basic MHT-encryption scheme.

5. ENCRYPTION BY MULTIPLE STATE INDICES IN THE QM CODER

The QM coder is an adaptive arithmetic coder that encodes binary symbol streams by using a very low cost probability estimation scheme. Unlike Huffman coding with fixed and predefined Huffman tree, the QM coder dynamically adjusts the underlying statistical model to a sequence of received binary symbols. The probability of binary symbols is updated via table look-up so that the price is very low. The QM coder is used in compression systems such as JPEG and JPEG2000. In this section, we propose a method to add encryption capability to the QM coder without affecting its compression performance or computational cost much.

In the original probability estimation machine of the QM coder, the state index is initialized to 0, which means symbols ‘0’ and ‘1’ are equally probable. Since there are only 113 possible values for this index, initializing it to a secret value provides no security. Therefore, we employ 4 indices, which are set to hidden initial values and used alternatively in a secret order to encode the input bitstream. The method for implementing encryption with multiple state indices (MSI) in the QM coder is called the MSI-coder. It consists of the following steps.

1. Generate a random key $K = \{(s_0, s_1, s_2, s_3), (p_0, \dots, p_{n-1}), (o_0, \dots, o_{n-1})\}$, where s_i is a 4-bit integer and p_i and o_i are 2-bit integers.
2. Initialize the four state indices (I_0, I_1, I_2, I_3) to (s_0, s_1, s_2, s_3) .
3. To encode the i_{th} bit in the input bitstream, we use index $I_{p_{(i \bmod n)}}$ to determine probability estimation Q_e . $I_{p_{(i \bmod n)}}$ is called the *active index*.
4. If the state update is required after encoding the i_{th} bit in the input stream, all state indices except $I_{o_{(i \bmod n)}}$ are updated.

Step 4 in the above procedure is employed to prevent four indices from synchronizing into the same values. Without step 4, if I_i and I_j happen to be equal at some time, they will always remain equal after that point. If this happens, the number of different state indices would be reduced, and the security of the scheme is compromised. In the extreme case where a very long run of 0’s or 1’s is inserted to the coder, all four state indices would become the same (equal to 112) and the MSI-coder is reduced to an ordinary QM coder. This “flooding attack” is also mentioned in previous work^{25,22} of integrating encryption with adaptive arithmetic coding with higher-order models as an effective attack method. The inclusion of Step 4 ensures that state indices will become different even if they are the same at some time point.

Image name	Total size (bits) of encoded DC coefficients using standard QM coder	Total size of encoded DC coefficients using MSI algorithm	Percentage of size increase
bike	260456	264944	1.72
cats	177240	183640	3.61
chart	173664	177536	2.23
cmpnd1	11728	12216	4.16
cmpnd2	1495504	1526608	2.08
gold	21656	22216	2.59
hotel	21824	22376	2.53
mat	62816	63328	0.82
seismic	10376	10528	1.46
tools	92880	84608	1.86
water	79632	80616	1.24
woman	230440	234288	1.67

Table 3. Comparison of the compression performance between standard QM coding and the proposed MSI-scheme

The computational cost in the algorithm for encryption is mainly contributed by Steps 3 and 4. Changing the active index in Step 3 costs about 2 CPU cycles. In Step 4, three indices are updated while the standard QM coder only requires one update. The two additional updates cost about 2 CPU cycles each so that the total computational cost is around 6 CPU cycles per bit encrypted. Another important issue is how the compression performance is effected by using multiple state indices. As shown in Table 3, the encoded data size of MSI-coder is only 0.82% ~ 4.16% larger than the original QM coder.

Finally, the security of MSI-coder could also be enhanced by the same methods adopted to enhance MHT-encryption in Section 4.2. The increase in computational cost caused by these enhancement methods is negligible.

6. CONCLUSION

A novel framework to protect confidentiality of multimedia data was introduced and two new encryption schemes were proposed to illustrate the framework in this research. Limitations of previous work on multimedia encryption were summarized. To overcome these problems, we proposed a new methodology of fast encryption by modifying the entropy coder in a multimedia compression system. Multiple statistical models were used alternatively in a secret order to encode the input symbol stream. Then, a fast encryption algorithm by using multiple Huffman coding tables was presented and two methods of enhancing its security were discussed. Another encryption scheme by using multiple state indices in the QM coder was also discussed. Both systems were shown to achieve security without much affecting the compression performance or the processing speed. There is more research to be done in the near future, for example, reducing the computational cost of the proposed MSI-scheme and refining the cryptanalysis for proposed algorithms.

REFERENCES

1. Intel, "Intel netstructure 7110 e-commerce accelerator fact sheet." www.intel.com.
2. H. Beker and F. Piper, *Secure Speech Communications*. New York: Academic, 1985.
3. B. Goldberg, S. Sridharan, and E. Dawson, "Design and cryptanalysis of transform-based analog speech scramblers," *IEEE Journal on Selected Areas in Communications*, vol. 11, pp. 735–744, June 1993.
4. S. Sridharan, E. Dawson, and B. Goldberg, "Fast fourier transform based speech encryption system," *IEEE Proc. I Commun., Speech, Vision*, vol. 138, no. 3, pp. 215–223, 1991.
5. C. J. Kuo and M. S. Chen, "A new signal encryption technique and its attack study," *IEEE International Carnahan Conference on Security Technology*, pp. 149–153, October 1991.
6. E. Dawson, "Design of a discrete cosine transform based speech scrambler," *IEEE Electronics Letters*, vol. 27, pp. 613–614, March 1991.

7. F. Ma, J. Cheng, and Y. Wang, "Wavelet transform-based analogue speech scrambling scheme," *IEEE Electronics Letters*, vol. 32, pp. 719–720, April 1996.
8. V. Milosevic, V. Delic, and V. Senk, "Hadamard transform application in speech scrambling," *IEEE International Conference on Digital Signal Processing*, vol. 1, pp. 361–364, July 1997.
9. B. Goldberg, S. Sridharan, and E. Dawson, "Cryptanalysis of frequency domain analogue speech scramblers," *IEEE Proceedings-I*, vol. 140, pp. 235–239, August 1993.
10. C. S. Xydeas, D. J. Hiotakakos, and C. A. Boyd, "Speech scrambling prior to lpc coding," *IEE Colloquium on Security and Cryptography Applications to Radio Systems*, pp. 9/1–9/4, 1994.
11. G. A. Spanos and T. B. Maples, "Performance study of a selective encryption scheme for the security of networked, real-time video," *Proceedings of the Fourth International Conference on Computer Communications and Networking*, October 1995.
12. G. A. Spanos and T. B. Maples, "Security for real-time mpeg compressed video in distributed multimedia applications," *15th IEEE International Phoenix Conference on Computers and Communications*, March 1996.
13. I. Agi and L. Gong, "An empirical study of secure mpeg video transmissions," *ISOC-SNDSS '96*, February 1996.
14. 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. Las Vegas.
15. 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. Boston.
16. L. Qiao, K. Nahrstedt, and I. Tam, "Is mpeg encryption by using random list instead of zigzag order secure?," *IEEE International Symposium on Consumer Electronics*, December 1997. Singapore.
17. L. Qiao and K. Nahrstedt, "Comparison of mpeg encryption algorithms," *International Journal on Computers and Graphics, Special Issue: Data Security in Image Communication and Network*, vol. 22, January 1998.
18. C.-P. Wu and C.-C. J. Kuo, "Fast encryption methods for audiovisual data confidentiality," in *SPIE Photonics East - Symposium on Voice, Video, and Data Communications*, (Boston, MA, USA), November 2000.
19. C. Shi and B. Bhargava, "A fast mpeg video encryption algorithm," in *Proc. of the sixth ACM international conference on Multimedia*, (Bristol, United Kingdom), September 1998.
20. C. Shi, S.-Y. Wang, and B. Bhargava, "Mpeg video encryption in real-time using secret key cryptography," in *Proc. of PDPTA '99*, (Las Vegas, Nevada), 1999.
21. A. Barbir, "A methodology for performing secure data compression," *Proceedings of the 29th Southeastern Symposium on System Theory*, pp. 266–270, March 1997.
22. H. A. Bergen and J. M. Hogan, "A chosen plaintext attack on an adaptive arithmetic coding compression algorithm," *Computers and Security*, vol. 12, pp. 157–167, 1993.
23. J. G. Cleary, S. A. Irvine, and I. Rinsma-Melchert, "On the insecurity of arithmetic coding," *Computers and Security*, vol. 14, pp. 167–180, 1995.
24. J. Lim, C. Boyd, and E. Dawson, "Cryptanalysis of adaptive arithmetic coding encryption scheme," *ACISP*, pp. 216–227, 1997.
25. I. H. Witten and J. G. Cleary, "On the privacy afforded by adaptive text compression," *Computers and Security*, vol. 7, pp. 397–408, 1988.
26. W. B. Pennebaker and J. L. Mitchell, *JPEG Still Image Data Compression Standard*. Van Nostrand Reinhold, 1993.
27. B. Schneier, *Applied Cryptography Second Edition : protocols, algorithms, and source code in C*. Wiley, 1996.