Design of Integrated Multimedia Compression and Encryption Systems

Chung-Ping Wu, *Member, IEEE,* and C.-C. Jay Kuo, *Fellow, IEEE*

***Abstract*— Two approaches for integrating encryption with multimedia compression systems are studied in this research, *i.e.* selective encryption and modified entropy coders with multiple statistical models. First, we examine the limitations of selective encryption using cryptanalysis, and provide examples that use selective encryption successfully. Two rules to determine whether selective encryption is suitable for a compression system are concluded. Next, we propose another approach that turns entropy coders into encryption ciphers using multiple statistical models. Two specific encryption schemes are obtained by applying this approach to the Huffman coder and the QM coder. It is shown that security is achieved without sacrificing the compression performance and the computational speed. This modified entropy coding methodology can be applied to most modern compressed audio/video such as MPEG audio, MPEG video and JPEG/JPEG2000 images.**

***Index Terms*— multimedia encryption, confidentiality, G.723.1 speech coding, CELP, Huffman coding, selective encryption, QM coder, signal scrambling.**

I. Introduction

The rapid growth of multimedia processing technologies and the wide availability of network access enable many powerful and creative new applications. Digital audiovisual contents can be created, edited, distributed, shared and stored with convenience at a low cost that has never been experienced before. Since the emerging wired and wireless IP networks are open networks, they are vulnerable to eavesdropping. Consequently, confidentiality is especially important for secure media distributions over IP networks. Internet telephony, Internet conferencing, Internet security monitoring and multimedia databases are a few examples of promising audiovisual data applications that require confidentiality. The development of fast and effective multimedia encryption technologies is crucial for IP-based multimedia applications to reach their full potential.

Conventional cryptographic schemes have been mainly developed for protecting alphanumeric data. Although encrypting the whole audiovisual data stream [1–3] with cryptographic ciphers is one way to achieve multimedia security, the large size of audiovisual data requires a considerable amount of computational power with this approach. Roughly speaking, audiovisual data usually contains a much lower information density than text data so that enciphering schemes with a significantly lower computational cost are feasible and desired. Moreover, the encryption/decryption speed is often critical to multimedia contents because real-time processing is required in some applications.

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 that 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 handheld 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 parallel. In fact, there are special servers [4] designed to handle encryption and decryption so that the main server could be freed from these computationally extensive operations. We may eliminate the need of a dedicated encryption server using a low complexity audiovisual encryption system.

Prior to the last decade, traditional solutions to audiovisual signal confidentiality were 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, low cost scrambling algorithms become vulnerable to attacks. More recently, the focus of multimedia security research shifted to integrating encryption with multimedia compression systems. The most popular method is selective encryption, where a portion of the coefficients from either the final results or intermediate steps of a compression system are enciphered with a cryptographic cipher. However, selective encryption is limited in its range of applications (see discussion in Section III). In the first half of this work, we examine limitations of selective encryption based on cryptanalysis and present two rules to determine whether this approach is suitable for a coded signal.

In the second half of this paper, we propose a novel multimedia encryption methodology that turns the entropy coder into a cipher using multiple statistical models alternately 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 side information. For ciphers, this information is the cipher key. For entropy coders, this information is the statistical model. 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 due to their low

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

computational cost, 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, we keep their basic structures unchanged but enlarge the statistical models used for symbol coding.

It should be noted that the encryption algorithms proposed in this paper aim at achieving confidentiality, not inhibiting unauthorized content duplication. The requirements of these two applications are different. Systems for inhibiting unauthorized content duplication attempt to prevent unauthorized users and devices from getting multimedia data with usable quality [5]. Such a system could be considered successful if the attacker without the correct key could only get highly degraded contents. Most selective encryption algorithms in the literature are adequate for this purpose. Encryption for confidentiality, on the other hand, must prevent attackers without the correct key from obtaining *any* intelligible data. Such a system fails if the hacker, after a lot of work, could make out a few words in the encrypted speech or a vague partial image from the encrypted video.

This paper is organized as follows. Traditional solutions to multimedia data confidentiality are reviewed in Section II. The limitations of selective encryption are examined and rules to determine whether this approach is suitable for a coded signal are proposed in Section III. The methodology of encryption via modifying entropy coding is discussed in Section IV. Two examples are given. One is based on multiple coding tables for the Huffman coder while the other uses multiple state indices for the QM coder. Finally, concluding remarks are provided in Section V.

II. Traditional Solutions to Multimedia Data Confidentiality

Most previous work in multimedia data confidentiality has been target at the speech signal. Early work [6] 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 [7], and are relatively easy to crack using modern computers [8]. As digital signal processing became popular, the focus shifted to scrambling in the domain of orthogonal transforms, such as DFT [8, 9], DCT [7, 10], the wavelet transform [11] and the Hadamard transform [12]. 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 permutations and/or affine transformations, they are very weak to known-plaintext [7] and chosen-plaintext attacks. Some successful plaintext-only attacks have also been reported [7, 13].

Furthermore, scrambling of the raw signal degrades the performance of multimedia compression systems, which have been designed based on the characteristics of unscrambled signals. While this problem could be alleviated by carefully designing the scrambling technique [9, 14, 15], deterioration in compression efficiency is inevitable. Since almost all multimedia data is compressed before transmission today, confidentiality schemes that are completely compatible with multimedia compression are more favorable.

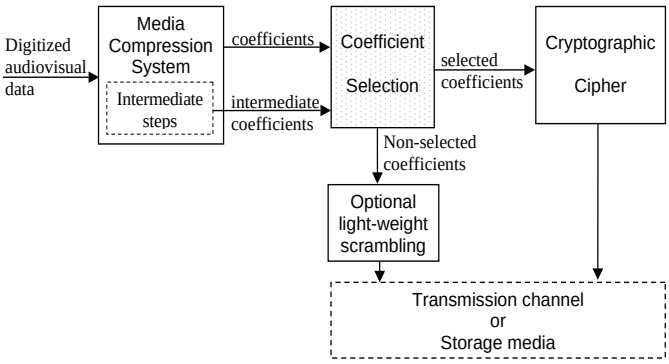


Fig. 1. The diagram of the selective encryption scheme.

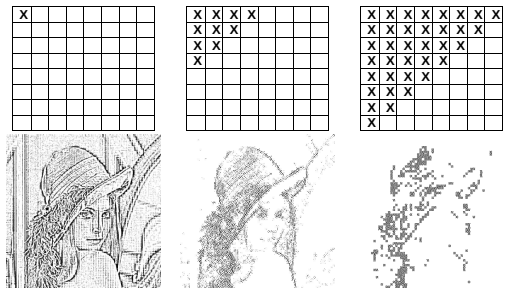
III. Selective Encryption Integrated with Multimedia Compression Standards

Most previous work on integrating encryption with multimedia compression standards to reduce the overall computation cost is focused on image/video data using some form of selective encryption. Although selective encryption is conceptually straightforward, many media types and compression systems are not suitable for this method. In this subsection, we will first review cryptanalysis of selective encryption algorithms in the literature, and present our cryptanalysis for the JPEG/MPEG sign-bit encryption scheme. Then, a rule-of-thumb for avoiding selective encryption is provided.

As shown in Figure 1, 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 AES or IDEA. The coefficients *not* selected are sent to the transmission channel (or storage media) with no encryption at all or only with light-weight scrambling. Since selected coefficients are protected by cryptographic ciphers, it is practically impossible for attackers to recover any information from these coefficients. If no intelligible information could be reconstructed from the remaining (non-selected) coefficients, the security level of the system is comparable to that of encrypting all coefficients. To be effective, the selected portion must be significantly smaller than the whole data stream.

A. Cryptanalysis for JPEG/MPEG Selective Encryption Schemes

Most existing selective encryption schemes aim 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. Early work in the field proposed to encrypt all I-frames in the MPEG video stream [16, 17]. It is argued that B frames and P frames cannot be reconstructed without I frames. However, Agi and Gong [18] showed that significant portions of the video are still visible under this scheme due to



(a) (b) (c)

Fig. 2. The effect of encrypting different numbers of DCT coefficients, where the 'X' mark indicates that the coefficient is encrypted and image restoration is done by setting encrypted coefficients to their mean value and performing image enhancement techniques such as equalization and thresholding manually. For hackers who are proficient with image processing tools, the image enhancement steps take around half an hour for each image.

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 $30\% \sim 60\%$ [18] or more [19] 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 that of total encryption. Moreover, identifying I-macroblocks in the MPEG stream introduces some computational overhead [19], which further decreases the gain of selective encryption.

Since both I-frames and I-macroblocks are entirely composed of DCT coefficients, researchers attempted to encrypt only some of DCT coefficients so that the 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, Tang [20] proposed a system that encrypts DC coefficients with DES and scrambles AC coefficients with block-based permutation. However, the *energy concentration* is often unrelated to the degree of intelligibility. Figure 2(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-plaintext [21] and chosen-plaintext attack. Since different AC coefficients possess different statistical characteristics, it was shown in [22] that low-resolution images could be reconstructed even using ciphertext alone.

Since encrypting only DC coefficients is not enough to destroy intelligibility, we further examine whether it would be helpful if a few low-frequency AC coefficients are also encrypted. Unfortunately, we see in Figure 2(b) that encrypting 9 AC coefficients has very little effect in destroying image edges, and even discarding more than half of the AC coefficients cannot fully destroy the image content.

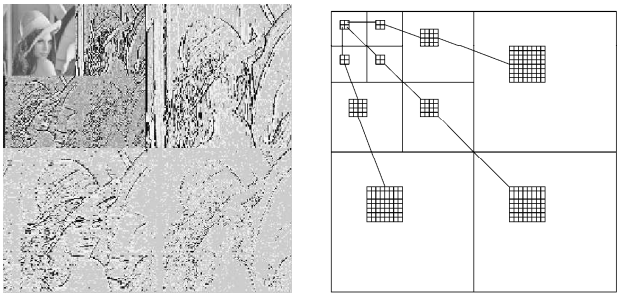
If selectively encrypting *some* DCT coefficients does not work, how about encrypting some part of *every* DCT coefficient? Shi and Bhargava proposed to encrypt the sign bit of every DCT coefficient in JPEG [23] and every motion vector in MPEG [24]. Although their experiments show that the image/video would be totally unrecognizable after sign bits are randomly altered, we are able to recover useful image contents by assigning all DC coefficients to 128 and all AC coefficients to *positive*. As shown in Figure 3 (a), most details of the Lena image are restored. The visual quality of recovered images in this figure is enhanced by using some basic image processing techniques such as sharpening, contrast adjusting and thresholding. For hackers who are proficient with image processing tools, performing image enhancement manually may take around half an hour for each image. It should be noted that a very vague Lena contour is recognizable even before the image enhancement task.

(a) (b) (c) (d)

Fig. 3. The effect of encrypting the most significant bits of every DCT coefficient under four cases: (a) the sign bit is encrypted; (b) two most significant bits are encrypted; (c) three most significant bits are encrypted; (d) four most significant bits are encrypted.

Since the sign bit could be viewed as the most significant bit of a coefficient, we experimented with further encrypting the 2nd, 3rd and 4th significant bits of every DCT coefficient. Figures 3 (b)-(d) show that the image reconstructed without the 4 most significant bits still reveals some information about the original image. Therefore, selectively encrypting the most significant bits of every DCT coefficient is not an effective way for image/video encryption.

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 Nahrstedt [19] 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. This method is only useful in applications where just a little speed improvement over total encryption would suffice. In order to achieve multifold speed-up, we may extend the concept and encrypt only one bit in every x bits, where $x \geq 3$. However, since this selection process ignores the issue of which bits are more important to intelligibility, it may not be a good selection policy and its security is questionable.



(a) (b)

Fig. 4. (a) Subbands of the wavelet transform of Lena image, and (b) a tree of wavelet coeffi cients.

B. Examples of Effective Selective Encryption Schemes

Although selective encryption schemes for JPEG/MPEG have not been very successful in achieving confidentiality, we do see some examples of effective selective encryption for other compression systems.

One such example is partial encryption for zerotree wavelet image compression proposed by Cheng and Li [25]. As shown in Figure 4(b), the wavelet transform for image creates a hierarchy of coefficients bands. Zerotree compression algorithms separate wavelet coefficients into significant and insignificant coefficients. The significance of a coefficient determines whether all coefficients further down the hierarchy tree will be coded. Cheng and Li proposed to encrypt the significance information related to the two highest pyramid levels. Hiding this information disrupts the determination of the tree structures, which is crucial to the decoding process. Therefore, although all subbands contain useful visual information as shown in Figure 4(a), it is extremely difficult to decode any of them without encrypted coefficients.

Another example is selective encryption for the CELP speech coder. The family of CELP speech codecs include popular standards such as ITU-T G.723.1, ITU-T G.729, GSM AMR, and MPEG-4 CELP. The CELP speech codec is model-based. That is, its coding principle is based on the analysis-by-synthesis method. The encoder incorporates a complete decoder unit, which synthesizes a segment of the output signal according to given input coefficients. The encoder systematically changes the coefficients until the difference between the original input signal and the synthesized output is within an acceptable range. The block diagram of the decoder is shown in Figure 5. The speech is synthesized using a model of the human voice generation mechanism as shown in Figure 6. The LSP decoder controls a linear filter, which models the changing shape of the human vocal tract. The pitch decoder models the periodic component of the excitation with the vibration frequency of the vocal cord, and the excitation decoder approximates the non-periodic component of the excitation signal.

We proposed a selective encryption algorithm for G.723.1 in [26]. Our analysis and experimental results indicated that the speech model in the decoder cannot function properly without several key coefficients in the G.723.1 bitstream, such as LSP codebook indices, the lag of pitch (pitch period) predictors,

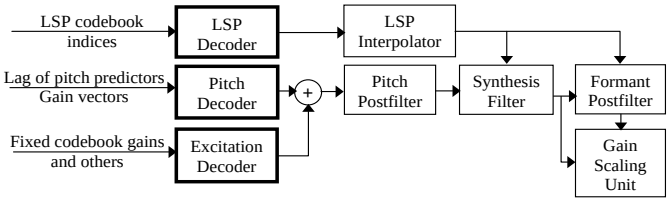


Fig. 5. The blockdiagram of the CELP speech decoder.

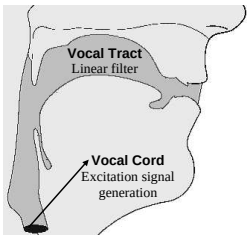


Fig. 6. The model of human voice generation mechanism, where the excitation signal generated by the vocal cord is modeled by the pitch and excitation decoder and the fi ltering effect of the vocal tract is modeled by the LSP decoder with a linear fi lter.

pitch gain vectors, and fixed codebook gains. We chose to encrypt the most significant bits of these coefficients, which consist of one-fifth to one-fourth of the whole bitstream. Servetti and Martin [27, 28] proposed a selective encryption system for G.729 by also selecting these four coefficients for encryption. They conducted formal listening tests, which showed that the reconstructed speech is completely intelligible.

C. Rules for Avoiding Selective Encryption

Based on the above arguments, we conclude that selective encryption is not compatible with multimedia compression systems possessing the following characteristics.

1) Based on an orthogonal transform followed by scalar 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. We have also observed similar results in MPEG audio compression. In contrast, the wavelet transform is followed by iterative thresholding and sorting to create zerotrees in the zerotree wavelet image compression scheme. Thus, hiding certain compression coefficients could inhibit the reconstruction of zerotrees and prevent any subband from being decoded.
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 (selective) encryption*" [9]. Similar statements can be found in [14, 20]. If selective encryption is performed after entropy coding, it is difficult to

distinguish which bits are more important as far as intelligibility is concerned.

Popular multimedia compression systems such as JPEG image compression, MPEG video compression and MP3 audio compression have both characteristics described above. They are transform-based schemes followed by scalar quantization and entropy coding. Thus, it is difficult to find an effective selective encryption scheme for them. In the next section, we present a new method for integrating encryption with these media compression systems as an effective means to achieve confidentiality.

IV. Encryption with Multiple Statistical Models in Entropy Coders

A. Background and Motivation

Since the entropy coder at the last stage of a media compression system could prohibit 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.

The encryption cipher and the entropy coder do bear some resemblance in that both of them turn the original data into redundancy-free bit streams, which cannot be decoded without certain information. For encryption, the information is the *key*; for entropy coding, the information is the *statistical model*. It is important to explore whether hiding this model could effectively prevent decoding of the compressed bit stream. Most previous work in this area has been focused on "adaptive arithmetic coding using higher-order models" [29–33]. 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 [29–32]. There seems to be no convincing remedies for these attacks yet [32]. However, 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 fact that an adaptive arithmetic coder with higher-order models adjusts its complicated statistical model after coding each symbol makes it a very slow coder. Consequently, it is not suitable for compressing large-sized audiovisual data.

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. Gillman *et al.* [34] showed that decoding a Huffman coded bitstream without any knowledge of the Huffman coding table would be tremendously difficult. This implies that an encryption system that uses the Huffman coding table as the *key* should be immune to ciphertext-only attacks, given that the table is not too small. However, it is obvious that this system is totally vulnerable to known-plaintext and chosen-plaintext attacks. Given a piece of the original data and its Huffman coded data, an attacker could determine many entries of the Huffman table by comparing the two data streams. When the length of the original data is long enough, the whole Huffman table could be resolved. Furthermore, in practical applications, the number of possible Huffman coding tables is often limited, and thus the key space is reduced. Shi and Bhargava [23] explored the possibility of encrypting MPEG video by hiding the Huffman coding table. To avoid affecting the compression ratio, not every different version of the Huffman coding table could be used. Consequently, even ciphertext-only attack by brute-force may be possible. It was concluded in [23] that hiding the Huffman table is not an ideal approach for MPEG video encryption.

The QM coder is an extremely simplified case of an adaptive arithmetic coder with higher-order models. Its initial state consists of only 3 integer numbers (A, C and $Q_e(S)$) [35]. Hiding these integers provides no security since 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 alternately 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 the key space increases with m^n, where n is the length of the hidden alternating sequence. The parameter m should be kept small since it is linearly related to the memory required to store these statistical models. Setting m to 8 and n to 128 generally produces a large enough key space for security.

In the next two subsections, 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.

B. Secure Huffman Coder with Multiple Coding Tables

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 the coding of each symbol is only a simple table lookup process, it is the fastest compression algorithm. In this section, we first present and analyze the basic algorithm of the multiple Huffman tables scheme (hereafter the MHT scheme). Then, two ways of enhancing the security of the MHT-encryption scheme are described.

1) Basic Algorithm: 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

[1]The concept of the MHT scheme was introduced in our previous work [26, 36].

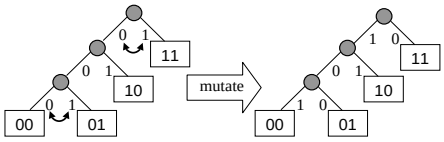


Fig. 7. Illustration of the Huffman tree mutation process.

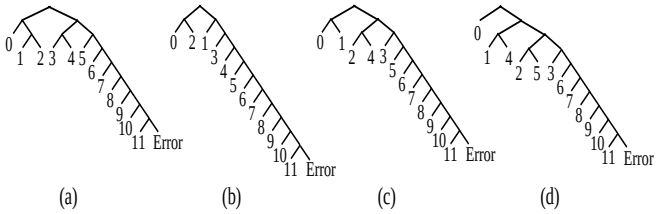


Fig. 8. The original Huffman coding tree for JPEG DC coeffi cient coding is given in (a) while three Huffman trees trained from three image sets are shown in (b), (c) and (d).

for decryption. The basic algorithm can be described in the following 3 steps.

1) Choose m different Huffman coding tables. They are numbered from 0 to $(m-1)$.
2) Generate a random vector $P = (p_0, ..., p_{n-1})$, where each p_i is a k-bit integer varying from 0 to $(m-1)$, and k equals to $\lceil \log_2 m \rceil$.
3) For the i_{th} symbol in the original data stream, use table $p_{i(\mathrm{mod}\ n)}$ 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 images (or audio pieces) in the world. If two Huffman tables happen to be identical, we can simply pick up one of them. With this approach, thousands of different Huffman tables can be obtained. In Step 1 of the algorithm, we may randomly choose m tables from the space of all available tables and send the indices as the key. If there is a total of 2^{N_t} published Huffman coding tables, each index would be N_t-bit long.

In the proposed system, we adopt an easier method to generate Huffman coding tables. Instead of training thousands of Huffman coding tables, we only train and obtained 4 different Huffman tables. Then, thousands of different tables can be derived using a technique called Huffman tree mutation. As shown in Figure 7 (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 7, we will get a new Huffman tree as shown in Figure 7 (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 t entries, its Huffman tree would have t leaves and $(t-1)$ inner nodes and label-pairs, which provides us the opportunity to make $(t-1)$ decisions about whether to permute each label-pair. Therefore, 2^{t-1} Huffman trees can be derived. In order to produce a new Huffman tree, we first randomly generate a $(t-1)$ 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 in reference to the original 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 8(a) shows the original Huffman tree used in encoding JPEG DC coefficients. When JPEG standard was made, this tree was created using the statistics collected from a set of images, which is assumed to properly represent all of the images that the JPEG system will encode. Figure 8(b)$\sim$(d) are three other Huffman trees that we trained with three independent image sets. The four Huffman trees in Figure 8 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, the total number of possible trees is $4 \times 2^{13-1} = 2^{14}$, *i.e.* N_t is equal to 14. Next, m of these possible Huffman trees are actually generated and used in the MHT algorithm as described above. As shown in Table I, 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, the MHT algorithm is better. The overall compression performance of MHT is approximately the same as that of the original Huffman coding.

Image name	Total size (bits) of encoded DC coeffs. with standard Huffman	Total size of encoded DC coeffs. 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 I
COMPRESSION PERFORMANCE COMPARISON BETWEEN THE STANDARD HUFFMAN CODING AND THE PROPOSED MHT-SCHEME.

It is possible to replace the original Huffman tree in the standard with other trees and sometimes achieve a smaller compressed data size because "Huffman coding with fixed and predetermined table" is inherently suboptimal in compression performance. The predetermined table is not tailored to the statistics of each image being compressed.

Table II 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 that basic tree. For example, the first column of Table II desribes the codeword lengths of any Huffman tree

Symbol name	Codeword length of Tree (a)	Codeword length of Tree (b)	Codeword length of Tree (c)	Codeword length of Tree (d)	No. of different 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 II

THE TOTAL NUMBER OF DIFFERENT CODEWORD LENGTHS FOR EACH SYMBOL WHEN THE FOUR HUFFMAN TREES IN FIGURE 8 ARE USED TOGETHER.

Cipher Type	Key-setup Cost (CPU instructions)	Encryption Cost (CPU instructions/bit)
MARS	9416	25
RC6	10372	22
Rijndael	35484	20
Serpent	26308	28
Twofi sh	37692	20

TABLE III

THE COMPUTATIONAL COSTS OF AES FINALISTS ON A PENTIUM-MMX MACHINE. THE FIGURES IN THIS TABLE ARE TRANSLATED FROM [37]BY ASSUMING 2 CPU INSTRUCTIONS ARE EXECUTED IN EVERY CLOCK CYCLE IN A PENTIUM-MMX CPU.

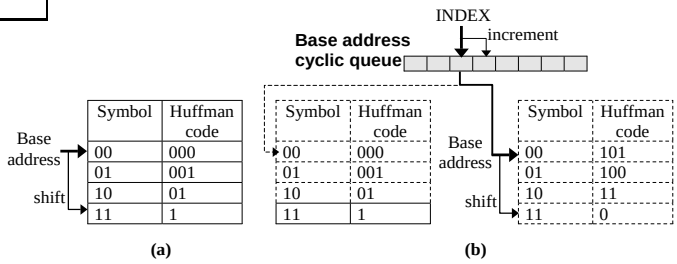


Fig. 9. (a) The normal Huffman coder adds the shift amount to the base address of the table to obtain the address of the desired Huffman code, and (b) the proposed algorithm loads the base addresses of Huffman tables from a cyclic queue, and the index to the queue should be increased by one after coding of each symbol.

mutated from Tree(a). As shown in Table II, when all Huffman trees are used alternately, usually 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.

2) Computational Cost: The evaluation of the computational speed of ciphers usually consists of the analysis of the key-setup cost, the encryption cost and the decryption cost [37]. The encryption and the decryption costs are usually similar, and they are more important than the key-setup cost because one single key-setup can often be followed by thousands of encryption/decryption operations. In the following, we analyze these costs of our MHT-encryption scheme, and compare them with those of modern ciphers.

1. Key-setup cost

The key-setup process includes all the computation and memory allocation operations prior to actual encryption of the first bit in the plaintext. The computational cost of MHT key-setup is dominated by the construction of multiple Huffman tables. Under a rough estimation of 20 operations per table entry, the total cost would be $20 \times t \times m$, where t and m are the table size and the number of selected tables, respectively. For the example of JPEG DC coefficient encryption as shown in the previous subsection, the key-setup cost would be around 2000 operations ($t = 13$ and $m = 8$). Compared with the ciphers listed in Table III, the key-setup cost of MHT-encryption is much smaller.

2. Encryption/decryption cost

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 9 (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 9 (b). Since each symbol in the original data usually corresponds to more than 3 bits in the Huffman bitstream, the net encryption cost of our algorithm is less than one CPU operation per encrypted bit, which is around 20 times smaller than the well-known AES finalists listed in Table III.

Recently, a new cryptographic cipher named COS [38, 39] with a very fast speed is gaining popularity. It is around $4 \sim 5$ times faster than AES. Compared to COS, the encryption cost of MHT is still several times smaller.

3) Cryptanalysis: Attack models are usually classified into three categories. In each model, every detail of the encryption/decryption algorithm is assumed to be known to cryptanalysts. The security depends solely on the hidden key.

1. Ciphertext-only attack

This is the least favorable situation for a cryptanalyst, where he only has the ciphertext to work with. Without any knowledge of the plaintext, a cryptanalyst usually analyzes the ciphertext by searching for statistical similarities between different pieces of ciphertext or some sequences that happen more than others. If the ciphertext is purely random without

statistical irregularities, he would have to resort to the brute-force exhaustive key search attack. We examine the pseudo-randomness of the ciphertext of the MHT-encryption scheme using the run length criterion proposed by Golomb [40]

$$R(l) = 2^{-l}r, \tag{1}$$

where r stands for the total number of runs in the sequence, and $R(l)$ is the number of runs of length l. A run is defined as a series of identical symbols (all zeros or all ones) which is preceded and followed by the other symbol. A pseudo-random sequence should be composed of runs whose statistics follow (1), *i.e.* the probability for a run to have length l is equal to 2^{-l}. Figure 10 shows the probability values of different run lengths in 3 ciphertext pieces, which are the encrypted DCT DC coefficients of 3 images from the JPEG2000 evaluation image set. The solid line represents the statistics of a perfectly random sequence (i.e. the formula in Equation 1). The probability values of the ciphertext under MHT-encryption, which are shown as circle marks, are very close to the ideal line in all test images. In contrast, the results of Huffman coding using only one coding table, as shown with cross marks, often deviate from the ideal line.

The strength of resisting exhaustive key search attack relies on a large key space. As explained earlier, a Huffman coding table with t entries could be mutated into 2^{t-1} different tables. Therefore, the 4 Huffman tables that are trained could produce $4 \times 2^{t-1}$ Huffman coding tables, from which m are chosen in the first step of our algorithm. The number of different ways to make this choice is $C_m^{4\times 2^{t-1}}$. Since the size of vector P is equal to n, there would be m^n different orders to use the Huffman coding tables. As a result, the size of the key space is

$$size(keyspace) = C_m^{4\times 2^{t-1}} \times m^n. \tag{2}$$

A practical Huffman coding table usually has more than 10 entries, which could mutate into at least 2^{10} different Huffman trees. When m is set to 8 and n is set to 128, the resulting size of key space would at least be

$$\text{size of the keyspace} = C_8^{4\times 2^{10}} \times 8^{128} = 2^{464.69} \tag{3}$$

It is practically impossible to perform brute-force search in a key-space of this size.

2. Known-plaintext attack

In this attack model, the cryptanalyst has a string of symbols in the original data and the corresponding encoded/encrypted bitstream. The original data could be obtained by inside-information or even by guessing. It is sometimes easy to make a correct guess. For example, most home video starts with the standard copyright warning screen.

Their goal is to find out which Huffman tables we have chosen and in what order they are used. First, they have to decide which bits in the bitstream correspond to which symbols in the original data. Since a symbol encoded with different Huffman tables would produce codewords with slightly different lengths, synchronization between plaintext and ciphertext is extremely difficult. For example, if each symbol has at least two possible Huffman codeword lengths, there would be more than 2^N possible ways to synchronize N symbols with the encoded/encrypted bitstream. Even in the case where each Huffman table has only about ten entries, at least a few hundred symbols need to be synchronized in order to reconstruct these Huffman tables and the ordering sequence. It is practically impossible to exhaustively try each synchronization pattern.

In the case where the cryptanalyst somehow found out the total number of bits in the ciphertext that corresponds to the N symbols, the possible number of synchronization patterns would decrease because not all codeword length combinations would result in the correct ciphertext length. In the following, we examine how many different ways of synchronization are left in this situation.

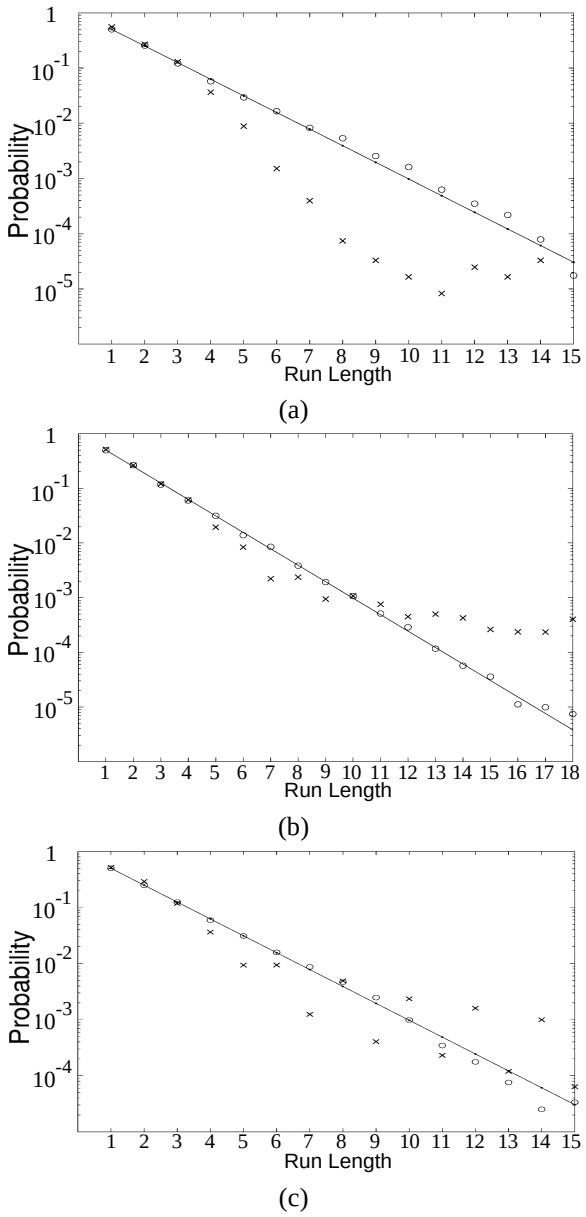


Fig. 10. The probability of different run lengths in the ciphertext of DCT DC coeffi cients of the (a) AERIAL2, (b) CMPND2 and (c) WOMAN image in the JPEG2000 evaluation image set. The circle mark stands for the result of MHT-encryption, and the cross mark represents the result of Huffman coding with a single coding tree.

Let us assume that each symbol s_i has two possible Huffman codeword lengths, l_i and l_i+1, and N symbols are known to map to $(L+\sum_{i=1}^{N} l_i)$ bits in the ciphertext, where L is in the range from 0 to N. Then, L of the N symbols will map to their longer codeword length, and the rest will map to their shorter codeword length. Thus, the number of possible ways of synchronization is C_L^N, which could be quite small when L or N is close to zero. However, the probability for this to happen is very small because it would require that all N symbols be simultaneously mapped to their shorter (or longer) length. In most cases, L will be close to $N/2$, therefore making C_L^N a very big number. For example, if N equals 100 and L is $N/2 \pm \delta$, C_L^N could be:

$$\begin{array}{ll} C_{50\pm\delta}^{100} = 2^{96.35} & \text{when } \delta = 0 \\ C_{50\pm\delta}^{100} = 2^{93.47} & \text{when } \delta = 10 \\ C_{50\pm\delta}^{100} = 2^{77.68} & \text{when } \delta = 25 \end{array} \tag{4}$$

The probability for δ to be larger than 25 is less than $2^{-22.32}$, which is small enough to ignore. Therefore, the total number of synchronization patterns in this example decreases from 2^{100} to $2^{77.68}$ in the worst case. The search space is reduced but still large enough to prevent exhaustive trial attack.

3. Chosen-plaintext attack

This situation is the most favorable to cryptanalysts. They could choose the plaintext and obtain the corresponding ciphertext. The goal is to obtain the secret key in order to decrypt the ciphertext to be encountered in the future. Our algorithm as described above is vulnerable to chosen-plaintext attack, especially when the attacker has the ability to insert one single symbol into the cipher and observe the corresponding output codeword. In this case, the attacker would have no synchronization problem at all. He could accumulate the symbol-codeword pairs and reconstruct the Huffman tables piece by piece.

In order to make synchronization more difficult, the proposed cipher should receive symbols as a whole chunk and output the corresponding codewords all together. The chunk size should be relatively prime to the size of vector P. To increase security, the chunk size should be as big as possible, but this will increase the coding delay. In the analysis of chosen-plaintext attacks, the attacker is usually assumed to have the power to reset the cipher as many times as wanted. With this ability, the attacker could perform differential analysis. For example, the MHT algorithm for JPEG DC coefficients as shown in Figure 8 and Table II could be used to illustrate this process. The attacker could encrypt the following two symbol chunks:

$$(0,0,0,\cdots,0) \ \ and \ \ (1,0,0,\cdots,0).$$

These two symbol chunks only differ by one symbol. The cipher is reset before encrypting each chunk, and the encoded bitstream sizes are recorded as x and y, respectively. If $y = x+1$, by examining the first two rows of Table II, the attacker would know that a tree mutated from Tree(a) must be the one used to encode the first symbol after the cipher is reset, and therefore the first symbol correponds to 2 bits of data in the ciphertext. Similarly, if $y = x+2$, the attacker would know that the first symbol is encoded into 3 bits of output data. The attacker could repeat this process for each symbol and each position in order to get synchronization between the plaintext and the ciphtertext.

The security analysis of the basic MHT-encryption scheme can be summarized as follows.

- The proposed scheme is resistant to ciphertext-only exhaustive key search attack due to a very large key space.
- 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.
- The basic MHT-encryption scheme is vulnerable to chosen-plaintext attack if the attacker could reset the cipher many times, encrypt a large amount of chunks of chosen-plaintext and compare the resulting ciphertext.

4) Security Enhancement: Based on the cryptanalysis given above, we present two methods to enhance the security of the basic MHT 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 basic MHT-encryption algorithm.

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 $Q = (q_0, ..., q_{\hat{n}-1})$, where each q_i is a 1-bit integer.
2) Perform function F_i on the $(w \times i)_{th}$ bit in the encrypted bitstream, $F_i(B) =$

$$\begin{cases} \text{do nothing}, & \text{when } q_{i(\bmod\, \hat{n})} = 0. \\ \text{add one random bit after } B, & \text{when } q_{i(\bmod\, \hat{n})} = 1. \end{cases} \tag{5}$$

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 low because bit-insertion is only performed once in every w bits.

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

The stream cipher is integrated into our system using 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 important that the same keystream is never used more than once [41]. The main computational load of stream ciphers is located in the keystream generator. It is very popular to use block ciphers such as AES 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

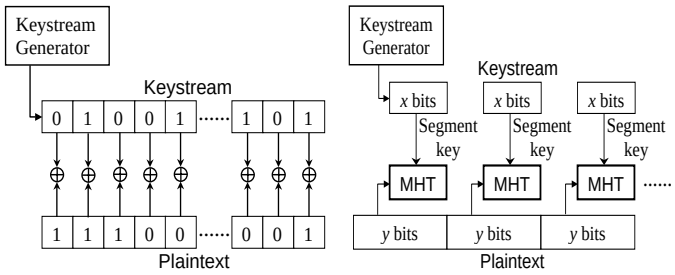


Fig. 11. (a) The structure of a standard stream cipher and (b) an encryption system combining a stream cipher with the MHT-encryption algorithm.

into segments of sizes 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 11(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 the keystream is used to encrypt y bits of the plaintext, the computational cost of this enhancement method is

$$C_{enh} = C_{str} \times \frac{x}{y}, \tag{6}$$

where C_{str} is the computational cost of a typical stream cipher. In order to make C_{enh} lower than the cost of the basic MHT-encryption scheme, which is about one CPU operation per bit encrypted, the segment size y should be determined as

$$\begin{aligned} & x = mN_t + n\lceil log_2 m \rceil \\ & \Rightarrow C_{enh} = C_{str} \times \tfrac{x}{y} = C_{str} \times \tfrac{mN_t + n\lceil log_2 m \rceil}{y} < 1 \\ & \Rightarrow y > C_{str} \times (mN_t + n\lceil log_2 m \rceil) \end{aligned} \tag{7}$$

C. Secure QM Coder with Multiple State Indices

The QM coder is an adaptive arithmetic coder that encodes binary symbol streams using a very low cost probability estimation scheme. Unlike Huffman coding with a 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 cost 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 the 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 alternately 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 $K = \{(s_0, s_1, s_2, s_3), (p_0, ..., p_{n-1}), (o_0, ..., o_{n-1})\}$, 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 $I_{p_{(i(mod\ n))}}$ to determine the probability estimation Q_e. $I_{p_{(i(mod\ n))}}$ 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 for $I_{o_{(i(mod\ n))}}$ 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 [30, 33] 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.

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 the MSI-coder. As shown in Table IV, the encoded data size of the MSI-coder is only $0.82\% \sim 4.16\%$ larger than the original QM coder.

Finally, the security of MSI-coder can also be enhanced using the same methods adopted to enhance MHT-encryption in Section IV-B.4. The increase in the computational cost caused by these enhancement methods is small.

V. Conclusion

Two approaches for integrating encryption with multimedia compression at a low computational cost were proposed in this paper. First, we discussed the benefits and limitations of selective encryption, and presented two examples of effective selective encryption schemes. We also conducted cryptanalysis for the JPEG/MPEG sign-bit encryption scheme and derived two rules for judging whether a multimedia compression system is suitable for selective encryption or not. Second,

Image name	Total size (bits) of encoded DC coeffs. with QM coder	Total size of encoded DC coeffs. with 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 IV
COMPARISON OF THE COMPRESSION PERFORMANCE BETWEEN STANDARD QM CODING AND THE PROPOSED MSI-SCHEME

for multimedia compression schemes that adopt the entropy coding at the final stage, we proposed a novel framework for fast encryption by modifying the entropy coder. Multiple statistical models were used alternately in a secret order to encode the input symbol stream. Two specific encryption methods were given. One is based on multiple Huffman coding tables to implement a secure Huffman coder while the other adopts multiple state indices to realize the secure QM coder. Two ways of security enhancement were discussed, and it was shown that security could be achieved without sacrificing the compression performance or the processing speed.

ACKNOWLEDGEMENTS

The research has been funded by the Integrated Media Systems Center, a National Science Foundation Engineering Research Center, Cooperative Agreement No. EEC-9529152. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the National Science Foundation.

REFERENCES

[1] D. Terry and D. Swinehart, "Managing stored voice in the etherphone system," *ACM Trans. Computer Syst.*, vol. 6, no. 1, pp. 3–27, 2 1988.
[2] L. Diez-Del-Rio *et al.*, "Secure speech and data communication over the public switching telephone network," *ICASSP-94*, Apr. 1994.
[3] J. Meyer and F. Gadegast, "Security mechanisms for multimedia data with the example mpeg-1 video," *Project description of SECMPEG, Technical Univ. of Berlin*, May 1995.
[4] Intel, "Intel netstructure 7110 e-commerce accelerator fact sheet," www.intel.com.
[5] M. Droogenbroeck and R. Benedett, "Techniques for a selective encryption of uncompressed and compressed images," *Proc., ACIVS*, Sept. 2002, ghent, Belgium.
[6] H. Beker and F. Piper, *Secure Speech Communications*. New York: Academic, 1985.
[7] B. Goldburg, S. Sridharan, and E. Dawson, "Design and cryptanalysis of transform-based analog speech scramblers," *IEEE J. Select. Areas Commun.*, vol. 11, no. 5, p. 735, June 1993.
[8] S. Sridharan, E. Dawson, and B. Goldburg, "Fast fourier transform based speech encryption system," *Commun., Speech and Vision, IEE Proc. I*, vol. 138, no. 3, p. 215, 1991.
[9] C. Kuo and M. Chen, "A new signal encryption technique and its attack study," *IEEE Int. Carnahan Conf. Security Technol.*, p. 149, Oct. 1991.
[10] E. Dawson, "Design of a discrete cosine transform based speech scrambler," *IEEE Electron. Lett.*, vol. 27, no. 7, p. 613, Mar. 1991.
[11] F. Ma *et al.*, "Wavelet transform-based analogue speech scrambling scheme," *IEEE Electron. Lett.*, vol. 32, no. 8, p. 719, Apr. 1996.
[12] V. Milosevic *et al.*, "Hadamard transform application in speech scrambling," *IEEE Int. Conf. Digital Signal Processing*, July 1997.
[13] B. Goldburg, S. Sridharan, and E. Dawson, "Cryptanalysis of frequency domain analogue speech scramblers," *Commun., Speech and Vision, IEE Proc. I*, vol. 140, no. 4, p. 235, Aug. 1993.
[14] C. Xydeas *et al.*, "Speech scrambling prior to lpc coding," *IEE Colloquium Security and Cryptography Applicat. to Radio Syst.*, p. 9/1, 1994.
[15] W. Zeng and S. Lei, "Efficient frequency domain selective scrambling of digital video," *IEEE Trans. Multimedia*, vol. 5, p. 118, Mar. 2003.
[16] G. A. Spanos and T. B. Maples, "Performance study of a selective encryption scheme for the security of networked, real-time video," *Int. Conf. Comput. Commmun. and Networking*, Oct. 1995.
[17] ——, "Security for real-time mpeg compressed video in distributed multimedia applications," *15th IEEE Int. Phoenix Conf. Comput. Commun.*, Mar. 1996.
[18] I. Agi and L. Gong, "An empirical study of secure mpeg video transmissions," *ISOC-SNDSS '96*, Feb. 1996.
[19] L. Qiao and K. Nahrstedt, "A new algorithm for mpeg video encryption," *First Int. Conf. Imaging Sci., Syst., Technol.*, July 1997.
[20] L. Tang, "Methods for encrypting and decrypting mpeg video data efficiently," *4th ACM Int. Conf. on Multimedia*, p. 219, Nov. 1996.
[21] L. Qiao *et al.*, "Is mpeg encryption by using random list instead of zigzag order secure?" in *IEEE Int. Symposium Consumer Electron.*, Dec. 1997.
[22] L. Qiao and K. Nahrstedt, "Comparison of mpeg encryption algorithms," *Int. J. Comput. and Graph.*, vol. 22, no. 3, Jan 1998.
[23] C. Shi and B. Bhargava, "A fast mpeg video encryption algorithm," in *6th ACM Int. conf. Multimedia*, Sept. 1998.
[24] C. Shi, S. Wang, and B. Bhargava, "Mpeg video encryption in real-time using secret key cryptography," in *Proc. PDPTA'99*, 1999.
[25] H. Cheng and X. Li, "Partial encryption of compressed images and videos," *IEEE Trans. Signal Processing*, vol. 48, p. 2439, Aug. 2000.
[26] C.-P. Wu and C.-C. J. Kuo, "Fast encryption methods for audiovisual data confidentiality," *SPIE Int. Symposium on Information Technologies 2000*, vol. 4209, p. 284, Nov. 2000.
[27] A. Servetti and J. Martin, "Perception-based partial encryption of compressed speech," *IEEE Trans. Speech Audio Processing*, vol. 10, p. 637, Nov. 2002.
[28] ——, "Perception-based selective encryption of g.729 speech," *Proceedings of ICASSP 2002*, vol. 1, pp. 621–624, May 2002.
[29] A. Barbir, "A methodology for performing secure data compression," *29th Southeastern Symposium Syst. Theory*, p. 266, Mar. 1997.
[30] H. Bergen and J. Hogan, "A chosen plaintext attack on an adaptive arithmetic coding compression algorithm," *Computers and Security*, vol. 12, p. 157, 1993.
[31] J. Cleary *et al.*, "On the insecurity of arithmetic coding," *Computers and Security*, vol. 14, p. 167, 1995.
[32] J. Lim, C. Boyd, and E. Dawson, "Cryptanalysis of adaptive arithmetic coding encryption scheme," *ACISP*, pp. 216–227, 1997.
[33] I. H. Witten and J. G. Cleary, "On the privacy afforded by adaptive text compression," *Computers and Security*, vol. 7, pp. 397–408, 1988.
[34] D. Gillman, M. Mohtashemi, and R. Rivest, "On breaking a huffman code," *IEEE Trans. Inform. Theory*, vol. 42, no. 3, p. 972, May 1996.
[35] W. B. Pennebaker and J. L. Mitchell, *JPEG Still Image Data Compression Standard*. Van Nostrand Reinhold, 1993.
[36] C.-P. Wu and C.-C. J. Kuo, "Efficient multimedia encryption via entropy codec design," *SPIE Int. Symposium on Electronic Imaging 2001*, vol. 4314, p. 128, Jan. 2001.
[37] J. Nechvatal *et al.*, "Report on the development of the advanced encryption standard," National Institute of Standards and Technology, U.S. Department of Commerce, Tech. Rep., Oct. 2000.
[38] E.Filiol and C.Fontain, "A new ultrafast stream cipher design: Cos ciphers," *8th IMA Conf. Cryptography and Coding*, Dec. 2001.
[39] E. Filiol. The cos cipher family. [Online]. Available: www-rocq.inria.fr/codes/Eric.Filiol/English/COS/COS.html
[40] V. D. Lubbe, *Basic Methods of Cryptography*. Press Syndicate of the University of Cambridge, 1998.
[41] B. Schneier, *Applied Cryptography Second Edition : protocols, algorithms, and source code in C*. Wiley, 1996.