© Copyright by Lintian Qiao, 1998

MULTIMEDIA SECURITY AND COPYRIGHT PROTECTION

BY

LINTIAN QIAO

B.S., Peking University, 1984
M.S., Marquette University, 1990

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1998

Urbana, Illinois

MULTIMEDIA SECURITY AND COPYRIGHT PROTECTION

Lintian Qiao, Ph.D.
Department of Computer Science
University of Illinois at Urbana-Champaign, 1998
Klara Nahrstedt, Advisor

A distributed multimedia system involves three components: the owner or the creator of the multimedia data, the user or the customer of the multimedia data, and in the middle, the network to deliver the multimedia data. This thesis is an effort to identify some of the open security problems related to these three components and to provide solutions to these problems.

In particular, we have designed and implemented the Encryption Algorithm, which utilizes the statistical behavior of MPEG compressed video. The statistical analysis of MPEG compressed video streams shows that the plain-text of compressed video has its own unique properties and is very different from text and other data types. This statistical analysis triggered our development of an efficient and secure algorithm for MPEG video streams. We introduced a watermark construction algorithm which applies a standard encryption function, such as DES, to the original multimedia data, and we have proved that this scheme is non-invertible and is successful against the "multiple claims of ownership" attack. We derived non-invertible watermarking methods which apply to image, video and audio streams. The watermark is embedded directly to the MPEG encoded bit streams to avoid the expensive three-step ("decoding-watermarking-encoding") process. We identified the customer's right problem and have designed two protocols to solve this problem. We discussed security issues within multicast environment and provided simple, efficient, and secure multicast protocols with copyright protection for the multimedia data.

To

my parents, Dongjun Qiao and Fengmin Fang,

and

my wife, Zhiying Xu

Acknowledgements

I would like to express my gratitude to all people who made this dissertation possible.

First and foremost, I would like to thank my advisor, Professor Klara Nahrstedt, for her dedicated supervision and guidance; for always encouraging me and being there when I needed help; and for giving me the freedom to explore. Every discussion and debate with her was helpful and improved significantly not only the quality of the research but also the clarity of this thesis.

I would also like to thank other members of my thesis committee: Professor William Kubitz, Jane Liu, and Shanghua Teng for their time and helpful comments.

I am grateful to Professor Saburo Muroga for initially encouraging and helping me to pursue the doctoral work.

Thanks are also due to all my colleagues and friends in the Multimedia Operating Systems and Networking (MONET) Research Group. Especially, I would like to thank Hao-hua Chu and Qiao Li for their friendship and many valuable discussions with them. Additionally, I give my thanks to the past and present MONET group members and friends from other research groups: Jian Li, Shigang Chen, Baochun Li, Daniel Lin, Zhigang Chen, Sergio Servetto, Srinivas Narayan, Alex Chen, Nancy Tran, Liman Wei, and Mingbiao Fei. My thanks also go to people who helped me cross many administrative hurdles: Anda Harney, Bonnie Howard, Barb Cicone, Julie Legg, and Erna Amerman.

Last, but by no means the least, my deepest gratitude goes to my parents, Dongjun Qiao and Fengmin Fang, for their encouragement, love, and support, and to my dear wife, Zhiying Xu, who always understands and supports me.

Table of Contents

Chapter

List of Tables

List of Figures

Chapter 1

Introduction

With the growth of multimedia systems in distributed environments, the problems associated with multimedia security and multimedia copyright protection have become a great concern to the public and to properly solve these problems is very important. Although computer and network security problems have been studied for a long time, many questions have been answered, and a lot of network security schemes and protocols are available, there are still open problems especially within the multimedia systems. This thesis is an effort to identify some of the problems and to provide solutions to these problems.

1.1 Motivation

Typically, a distributed multimedia system involves three components: the owner or the creator of the multimedia data, the user or the customer of the multimedia data, and in the middle, the network to deliver the multimedia data. For these three components, different issues related to the security and copyright exist:

- we must provide secure network environment,
- we need schemes to protect owner's copyright,
- we must consider the user's customer-rights.

In order to provide secure data transmission, it is necessary to design special encryption algorithms for multimedia data because of their special characteristics, such as their coding structure, large amount of data, and real-time constraints. In particular, we are interested in the secure algorithms for MPEG video streams. The MPEG video encryption algorithms should aim towards efficient and real-time processing so that they can become an integral part of the video delivery process and at the same time preserve the highest security level and compression ratio. There already exist several encryption algorithms to secure MPEG video. However, each of them has its weakness in terms of either security level, or speed, or resulting stream size metrics.

Digital watermarking techniques have been proposed in recent years as methods to protect the copyright of multimedia data. However, there are still open problems. One of them is how to resolve the *rightful ownership of the invisible watermarking schemes*. The rightful ownership problem is either not addressed at all or not addressed properly within current existing watermarking techniques. Furthermore, although the concept of *non-invertibility*, which means that a watermarked image/video/audio

- can't be easily stripped off of its watermark,
- can't be easily falsified,
- is properly protected from falsifying the used watermark or/and watermarked data,

has been defined to resolve the rightful ownership, the previously proposed non-invertible scheme cannot be proved to be non-invertible. Meanwhile, different data types, such as image, video, and audio, must be handled differently because each of them has its own special properties such as size, compression property, and sensitivity to the distortion.

Another open problem refers to the customer's right protection. The *rights of the legitimate customers/buyers* in using watermark have not been addressed at all. Intuitively, the watermark schemes can be used to identify different customers. However, this depends on our trust to the owner (or the party to embed the watermark) because the owners have the total control of the watermarking process. We shall see that we can not always automatically trust the owner. Without properly solving the customer's right problem, the goal of using watermark to protect the owner's copyright can also be defeated.

The security issues within multimedia multicasting environment are also very interesting. Multicasting schemes try to share the transmitted data between different users, on the other hand, security and copyright protection require differentiation of the users and have to avoid sharing. These two opposite natures make security and copyright protection in multicasting a very difficult problem.

1.2 Thesis Outlines

In this thesis, we discuss various security issues in details and present solutions to the problems described in previous section. In particular:

-) We have designed and implemented the *Video Encryption Algorithm (VEA)*, which utilizes the statistical behavior of MPEG compressed video. The statistical analysis of MPEG compressed video streams shows that the plain-text of compressed video has its own unique properties and is very different from text and other data types. This statistical analysis triggered our development of an efficient and secure algorithm for MPEG video streams.

-) We present watermarking methods which resolve the rightful ownership and are successful against the "multiple claims of ownership" attack. We introduce requirements on the watermark construction algorithm which include (1) using a standard encryption function such as DES [46], and (2) using the original image/video or other known bit stream information during the watermark construction process. Because of these requirements, the non-invertibility of our proposed scheme is easily proved. Although our methods are mainly applied to MPEG-encoded video streams, they can be easily extended to many image schemes or other video schemes. We apply our scheme to raw images and prove its non-invertibility property.

-) We present non-invertible watermarking methods which applies to MPEG audio streams. The watermark is embedded directly to the MPEG audio bit streams to avoid the expensive three-step ("decoding-watermarking-encoding") process when watermarking schemes designed for uncompressed audio are applied to MPEG audio. Meanwhile, because sound distortion is more sensitive to human ear than visual distortion to human eye, special consideration is taken in designing such schemes.

-) We design two protocols to provide protection of customers' rights. We introduce a certificate/identification information for the customer. The customer's certificate needs to be tied into the watermarking process at the owner's side. Only the customer knows the key which creates the certificate. Once the customer purchases the watermarked data, he/she can prove his/her rights by showing that the certificate is indeed created by his/her private key. The secure exchange of the certificate and watermarked data is performed by our protocols.

-) We present a simple, efficient, and secure multicast protocol targeting to MPEG video streams within an open and insecure network environment. (1) It does not require any security mechanism on intermediate network switches or routers. (2) It can run on top of any existing multicast architectures. (3) It is robust in the presence of long message delay for membership update. (4) It can handle dynamic group membership, e.g., join/leave/expel, in a bandwidth and time efficient manner.

Figure 1.1 shows a typical distributed multimedia system with an emphasis on the security issues.

The thesis is outlined as follows: Chapter 2 presents the related work in the area of multimedia security methods. It includes topics in regular data encryption algorithms, network security protocols, MPEG encryption algorithms, watermarking schemes, authentication and internet security issues, and security studies in multicasting environment. Chapter 3 provides attacks on existing MPEG encryption algorithms, presents the statistical analysis results on the MPEG video streams, and proposes our Video Encryption Algorithm. Chapter 4 discusses the rightful ownership problem, formally defines the non-invertibility property of using watermark, and presents our watermarking methods which resolve the rightful ownership. Chapter 5 provides necessary background for MPEG audio compression, designs the non-invertible watermarking schemes for MPEG audio, and presents our experimental results. Chapter 6 identifies the customers' right problem and proposes our customers' right protocols. Chapter 7 discusses various security issues within multicasting environment and presents the secure multicasting protocols for MPEG video streams. Finally, research accomplishments and future work are summarized in Chapter 8.

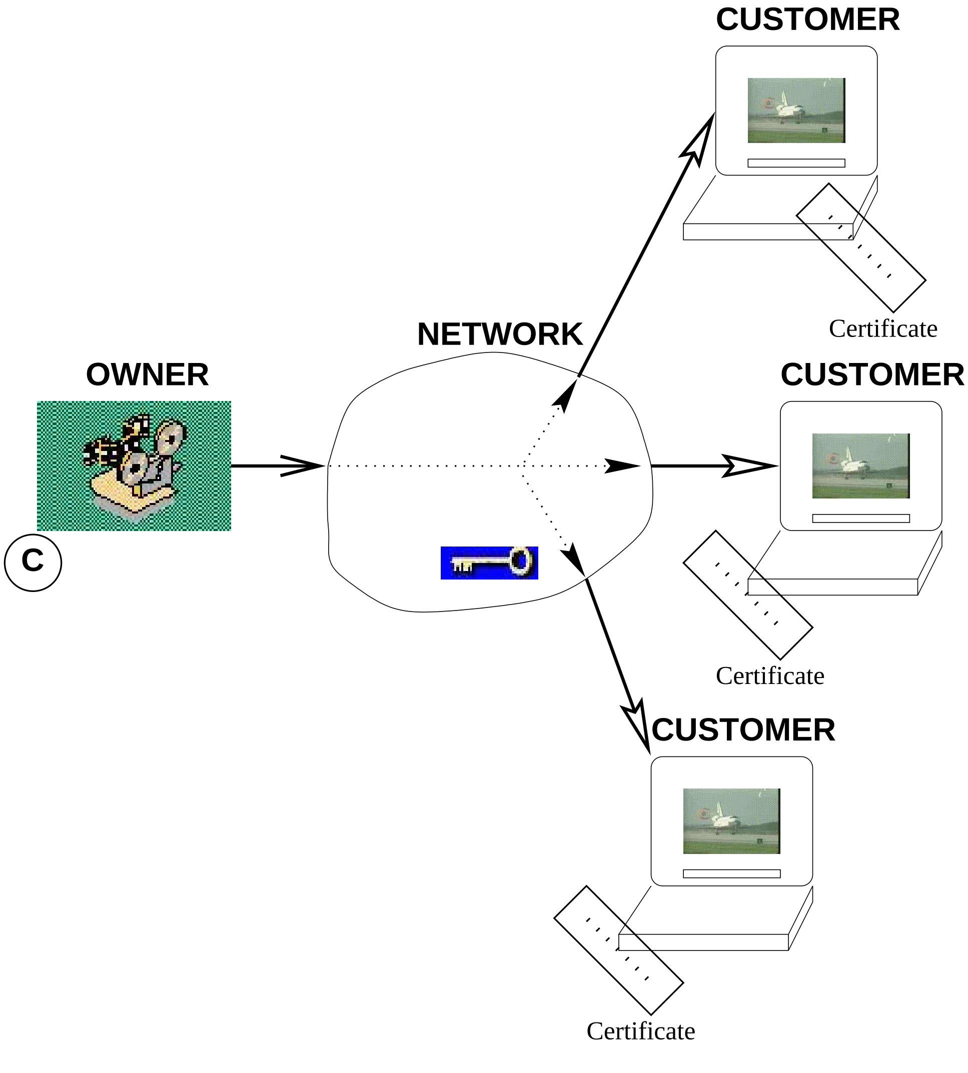

Figure 1.1: Security Issues in Distributed Multimedia System.

Chapter 2

Related Work

2.1 Data Encryption Algorithms

There are two basic technologies for encryption and decryption: Secret Key Encryption and Public Key Encryption [24]. DES, the Data Encryption Standard which was defined by the US government in 1977, is a secret-key, symmetric cryptosystem. The standard DES operates on 64-bit blocks with a 56-bit key and is basically a 16-round Feistel cipher (16 rounds of splitting, permutation, and xor). Longer key size and triple encryption DES have been suggested in order to enhance the security against possible attacks (e.g., differential cryptanalysis [12]). IDEA (International Data Encryption Algorithm [35]) is another encryption algorithm which is considered to be immune from differential cryptanalysis. IDEA has a similar speed of DES. RSA [53] is a public-key cryptosystem for both encryption and authentication (developed by Ron Rivest, Adi Shamir, and Leonard Adleman in 1977). The problem with the known public-key encryption method is the speed. It is slower than the secret key scheme.

2.2 Secure Transmission for Multimedia Data

A few methods dealing with multimedia data were developed. Macq and Quisquater [38] developed a multiresolution encryption algorithm where the low-resolution information of the images (i.e. their icons) remains unencrypted. Yamamoto [68] describes an integration of PGP and MIME (Multipurpose Internet Mail Extensions) so that the security for non-textual objects in MIME format can be supported. However, the algorithm is simply applying PGP [70] to MIME.

Some proposed attempts to secure MPEG streams have been reported. The most straightforward method is to encrypt the entire MPEG stream using standard encryption methods. This is called the *naive algorithm* approach [2]. The greatest concern about this approach is the speed of processing due to the large size of MPEG files.

Another method to secure MPEG streams is the *selective encryption* algorithm which encrypts only the I-frame of MPEG streams ([39], [36]). Meyer and Gadegast have designed a new MPEG-like bit-stream SECMPEG that incorporates selective encryption and additional header information, and has high-speed software execution [42]. SECMPEG can use both DES and RSA and implements four levels of security: 1st level – encrypts all headers. 2nd level – encrypts all headers plus the DC and lower AC terms of the I-blocks. 3rd level – encrypts I frames and all I-blocks in P and B frames. 4th level – encrypts all data. SECMPEG is not compatible with standard MPEG. A special encoder/decoder would be required to view unencrypted SECMPEG streams.

A proposal targeting at integration of compression and encryption of MPEG streams into one step is presented in [62], where the basic idea is to use a random permutation list to replace the zig-zag order to map the individual 8x8 block to a 1x64 vector. We call this method the "ZigZag-Permutation Algorithm".

2.3 Watermarking Schemes

During the past few years, a number of digital watermarking methods [22] have been proposed. Among the earliest works, L.F. Turner [63] proposed a digital audio watermarking method which substitutes the least significant bits of randomly selected audio samples with the bits of an identification string (watermark). A similar idea can also be applied to images [64]. However, because of the use of least significant bits, the identification code can be easily destroyed. There are many other proposals for watermarking such as Tanaka's schemes [61] which use the fact that the quantization noise is typically imperceptible to users, Brassil's methods [15] for textual document images, and Caronni's geometric patterns (also called tags) [17]. These schemes are easily defeated by filtering, redigitization, geometric distortion, cropping, requantization, etc.

Bender, Gruhl, and Morimoto [11] discuss two different approaches to watermarking: (1) *Patchwork*, and (2) *Texture Block Coding*. Patchwork is a statistical approach. It randomly chooses a pair of image points (a_i, b_i), then increases the brightness of a_i by one and decreases the brightness of b_i by one. These two steps are repeated n times. The expected value of the sum of the difference of such n pairs is then 2n. This method is resistant to compression and FIR filters and remarkably resistant to cropping. However, its assumption that all brightness levels are equally likely is not practical. Texture Block Coding is a visual approach. It hides data in continuous i random texture patterns. The texture blocking scheme is implemented by copying a region from a random texture pattern found in a picture to an area which has a similar texture. The auto-correlation of the image results in the recovery of the shape of the region. This method is limited to those images with a lot of textures.

Smith and Comiskey [56] discuss the characteristics of the data hiding schemes, such as the amount of information that can be hidden, the perceptibility, the robustness which is modeled using the quantities of channel, the signal-to-noise ratio, and the jamming margin. They introduce new data hiding schemes whose parameters can be adjusted with the capacity, imperceptibility, and robustness.

Koch, Rindfrey, and Zhao [26] present an image watermarking method. The image is grouped into 8x8 blocks and each 8x8 block is DCT transformed. They randomly choose a subset of those blocks, and for each block in that subset, one of the 18 possible triples of frequencies is selected and modified so that their relative strengths encode as 0 or 1. However, because the choice of the frequency set is not based on any perceptual significance and the variance of the frequency coefficients is small, this method may be sensitive to noise or distortions. It is also vulnerable to *multiple-document attacks*.

Cox, Kilian, Leighton, and Shamoon [21] argue that a watermark must be placed in perceptually significant components of a signal if it is to be robust to common signal distortions and attacks. To avoid the perceptual degradation because of watermarking those components, they propose to insert a watermark into the spectral components of the data using techniques analogous to spread spectrum communications, hiding a narrow-band signal in a wide-band. The watermark consists of 1000 randomly generated numbers. The length of the watermark is variable and can be adjusted to suit the characteristics of the data.

Schneider and Chang [52] present a methodology for designing content based digital signatures which can be used to authenticate images. Using their methodology, signature systems can be designed which allow certain types of image modification but prevent other types from manipulation. They also propose an extension of the authentication system towards video.

Langelaar *et al* [27] propose two image watermarking methods. The first one extends the existing spatial labeling technique which adds a positive integer constant k to the brightness of 50% of the pixels in an image. This constant k is called the label embedding level. By dividing the image into blocks and searching an optimal label-embedding level k for each block instead of using a fixed embedding level, a large and more robust label can be embedded in an image. The second method removes high frequency DCT-coefficients in some areas to embed a label. However, this method may remove too many DCT-coefficients therefore cause distortions.

Hartung and Girod [34] provide a watermarking scheme specifically for MPEG encoded video. The basic idea is to apply DCT to each of 8x8 blocks of the watermark and then add the DCT coefficients of the watermark to the corresponding DCT coefficients of MPEG stream. However, if the length of the new Huffman codeword is longer than the old one, then the algorithm restores the old DCT coefficient of the MPEG stream. This method guarantees that the length of watermarked video stream does not increase. The problem is that it may violate the requirement of labeling most significant components. The watermark embedding process is also slow. It can only handle data at a rate of several bytes per second.

Craver, Memon, Yeo, and Yeung [23] address an important issue of rightful ownership. They provide counterfeit watermarking schemes that can be performed on a watermarked image to allow multiple claims of rightful ownerships. We will refer to this attack as the *CMYY attack*. We will discuss this topic in detail in the next section.

Boney, Tewfik, and Hamdy [14] present a technique for embedding watermarks into audio signals. In their scheme, both the perceptual masking models of the HAS (Human Auditory System) and the temporal masking models are used. In particular, the watermark is generated by filtering a PN-sequence (Pseudo-Noise sequence) with a filter that approximates the frequency masking characteristics of the HAS. It is then weighted in the time domain to account for temporal masking.

Dittmann, Nack, A. Steinmetz, and R. Steinmetz [25] discuss the acceptance problems and risks of existing watermark techniques in robustness, quality loss, and lack of visual interactive handling support of watermark embedding and retrieval. They also describe four prototype tools as parts of an interactive multimedia watermarking environment for digital data: the two LabellingEditors for images and video, the 3D-watermark, and the ObjectTrackingEditor.

However, we need to point out that all of the schemes discussed in this section can not sustain against the CMYY attack therefore fail to resolve the rightful ownership.

2.4 General Network Security Issues

In order to apply watermarking schemes properly within distributed environment, other security problems must be considered, such as key distribution, authentication, non-repudiation of multimedia data, secure electronic payments, and so on. In this section, we consider currently available techniques in these areas.

In 1988, Steiner, Neuman and Schiller [57] from MIT designed Kerberos. Kerberos is a network authentication system used on physically insecure networks. It is based on the key distribution model presented by Needham and Schroeder [45]. Bellovin and Merritt [10] discussed a number of limitations and weaknesses of the Kerberos system and presented solutions to some of them.

Recently, there has been works on the principles and general requirements of cryptographic protocols. For example, Abadi and Needham [1] presented several principles on designing security protocols. Although these principles are neither necessary for correctness, nor are they sufficient, they can be used as informal guidelines and are useful independently for any logic. Studies on general requirements and formal methods to analyze security protocols can be found in [[16], [59], [13], [60], [32], [58]].

To answer the question of "When was a record created or last modified?", digital timestamping systems are proposed. Such systems are described in [8] [33]. A digital timestamping system has a certification procedure which the author of a record can "sign" the record, and a verification procedure by which any user can check a record and its accompanying certificate. The verification procedure takes a particular record and a putative timestamp certificate for that record and a particular time, and uses this information to validate whether that record was indeed certified at the time claimed by checking it against the widely available summary number for that moment.

Reiter and Stubblebine [50] discussed the concept of authentication metrics, a procedure for evaluating the assurance one has in a name-to-key binding. They developed a number of principles for the design of authentication metrics, demonstrated why several proposed metrics fall short, and described a new metric that comes close to being an acceptable metric of authentication.

Franklin and Reiter [30] provided a fair exchange protocol which enables two parties to exchange secrets with fairness so that neither of the parties can gain an information advantage by quitting prematurely or otherwise misbehaving. A "semi-trusted" third party is used to facilitate the exchange. Asokan, Schunter, and Waidner [3] presented optimistic protocols for fair exchange of electronic data with non-repudiation. These protocols do not involve a third party during a fault-less exchange but do require one during their recovery. Zhou and Gollmann [69] also presented a fair non-repudiation protocol which requires a Trusted Third Party but the involvement of the Trusted Third Party is minimized.

Gong [31] developed techniques which construct authentication protocols on the basis of one-way functions rather than encryption algorithms. Equally simple and capable protocols were achieved in comparison with either public-key or private-key encryption algorithms.

Bellare *et al* [9] proposed a family of protocols - iKP - for secure electronic payments over the Internet. The protocols involve three parties: the customer who makes the payment, the merchant who receives the payment, and a gateway which connects the electronic world with the existing payment infrastructure and authorizes the transactions. All iKP protocols are based on public-key cryptography and offer different levels of security.

Cox, Tygar and Sirbu [20] [55] presented NetBill security and transaction protocol. The features of the protocol include: (1) an atomic certified delivery method so that a customer pays if and only if he/she receives his/her information goods intact; (2) outsourcing access control – different users can use different access control servers; (3) a credential mechanism allowing users to prove membership in groups; (4) a structure for constructing pseudonyms to protect the identities of consumers.

Another interesting topic involving network commerce is providing anonymity and untraceability against sellers. This is in the opposite direction of applying watermark whose purpose is to identify the sellers. Such systems can be found in *ecash* and *CAFE* [28] where both depend on blind signatures [18]. *NetCash* [40] [41] and *ACC* [37] also provide anonymity and untraceability, but they are based on the use of trusted third parties that change electronic money of one representation into another representation, without revealing the relation.

2.5 Security Issues within Multicast Environment

There are basically 2 major approaches on how the multicast routers can multicast – the "soft state" vs. "hard state". The soft state approach, e.g., Multicast Open Shortest Path First Protocol (MOSPF) [44] and Distance Vector Multicast Routing Protocol (DVMRP) [65], does not require the multicast routers to keep states of the multicast connections (e.g. the distribution tree and locations of the members). On the other hand, the hard state approach, e.g., Core Based Tree (CBT) [4, 7] and Protocol Independent Multicast (PIM) [29], requires the multicast routers to maintain the states of the multicast connections.

Almost all multicasting security schemes are focused on the problem of key management. The goal of the key management is to distribute the *group key* securely to the group members, who can then use it to encrypt or decrypt the multicast data. They deal with scalability issues in terms of group size. We will describe most prominent schemes, such as the *Complementary Variable Approach*, the *Hierarchical Tree Approach*, *CBT Key Distribution Approach*, and *Iolus Framework.*

First, we describe the *Complementary Variable Approach.* let us assume that the root and each leave have a pairwise KEY_i. This KEY_i is used by root to distribute the Net Key (It is also called group key. Here we follow the notation used by Wallner et. al). In addition, root also creates n "complementary variables". Leaf i will receive all complementary variables except its own. To cut a leaf, for example, leaf i from the group, all other leaves and root generate a new Net Key based on old Net Key and i-th complementary variable. One way to do this is to use the old Net Key and i-th complementary variable as a seed to create a random number sequence. The created random number sequence is the new Net Key. Only leaf i can not get this new Net Key because i does not know the i-th complementary variable. The major attack on this scheme is that when cutting multiple leaves from the group, these deleted users could work together and compute the new Net Key by sharing their complementary variables.

Next, we discuss the *Hierarchical Tree Approach.* Each multicast session has a key server which maintains a tree (from the bottom up) with each leaf corresponding to each user. Each user establishes a unique pairwise key with the server. Once each user has a pairwise key known to the server, the server generates a key for each remaining node in the tree. Starting with those nodes all of whose children are leaves and proceeding towards the root, the server transmits the key for each node, which encrypted using the child-node's keys. At the end of the process, each user can determine the keys corresponding to those nodes above her leaf. In particular, all users hold the root key, which serves as the common Net Key for the group. This scheme can efficiently remove a user from the group (for a multicast session with n users, log(n) new keys are needed). However, this algorithm does not address the issue of multiple-user "join/leave/expel" operations. In the worst case, if we want to remove every other users from the session, the total number of new keys needed is 2n.

CBT Key Distribution is based on a "hard state" multicast protocol like the CBT. The tree is permanently created on the multicast routers, and the routers can maintain various security information, e.g., the access control list (ACL), and the group key, in its hard state. The efficiency of this approach is in the key distribution algorithm. Because a centralized approach is not scalable, it tries to improve upon a centralized key distribution approach by doing key distribution on the routers. This means that after the core router authenticate as non-core router, the core router passes the ACL and group key (using asymmetric encryption) to the non-core router and the non-core router becomes trusted. With the ACL, the non-core router can also authenticate other hosts or routers. ACL (access control list) is used to authorize hosts to join the multicast group. It is given by the initiating host. There is a strong assumption that all the ROUTERS can be trusted and they do not misbehave. It also does not address dynamic host "join/leave" behavior.

Suvo Mittra proposes Iolus framework for scalable secure multicasting based on the notion of a secure distribution tree. Iolus considers the member join/leave operations and suggests to multicast n copies of the subgroup key (assumption is that there are n members in the subgroup), each copy is encrypted with the different member's secret key. However, it is not clear how to deliver the keys in a timely fashion to meet the time constraints of multimedia data. Also the parent of a subgroup must (1) decrypt the sender's message, (2) re-encrypt the message, and (3) send it out to the upper level. This is not efficient and may have security breach too.

All schemes above do not take the long network delay or message loss into consideration. When some senders of a multicast session do not get the rekey message in time or even never get the rekey information due to the message loss, they may continue to multicast data use old group keys. Expelled users can still decrypt multicasted data. This may create serious security problem.

Chapter 3

MPEG Video Encryption Algorithm

3.1 Introduction

Emerging distributed multimedia applications such as Video-on-Demand, video broadcast, multimedia mail and video conferencing must be provided with secure transmission. One approach is to simply apply known encryption algorithms such as DES [46] or RSA [53]. However, if a user wants to use the encryption as part of the video retrieval and playback process in real-time, DES and RSA are not fast enough to satisfy this requirement. Hence, there is a need for a new secure and fast encryption algorithm for compressed video streams.

We have designed and implemented a new algorithm, called the *Video Encryption Algorithm (VEA)*, which utilizes the statistical behavior of a compressed video. The statistical analysis of compressed video streams shows that the plain-text of compressed video has its own unique properties and is very different from text and other data types. This statistical analysis triggered our development of an efficient and secure VEA.

3.2 Background – MPEG Video

Let us first briefly review the MPEG compression algorithm. The basic idea behind MPEG video compression is to remove spatial redundancy within a video frame and temporal redundancy between video frames. It uses Discrete Cosine Transform (DCT) method to reduce the spatial redundancy and uses motion-compensation to exploit the temporal redundancy.

In the first step, the RGB format image is converted to YUV format in which a pixel is represented with 24 bits (8 bits each for Y, U, and V). U and V are further subsampled to 2 bits each and the total bits reduced to 12, thus it automatically gives 2:1 compression ratio.

Frames are divided into 16x16 pixel macroblocks which contain four 8x8 Y blocks and two 8x8 U and V blocks. The blocks are transformed using the DCT method and then the coefficients are quantized, zig-zag scanned and run-length coded to remove redundant zeros, and entropy coded (using Huffman coding). As a result, I frame will be very similar to JPEG format. The first DCT coefficient of each block is called DC coefficient and the rest are called AC coefficients. The zigzag ordering of the coefficients in one 8x8 block is shown in Figure 3.1.

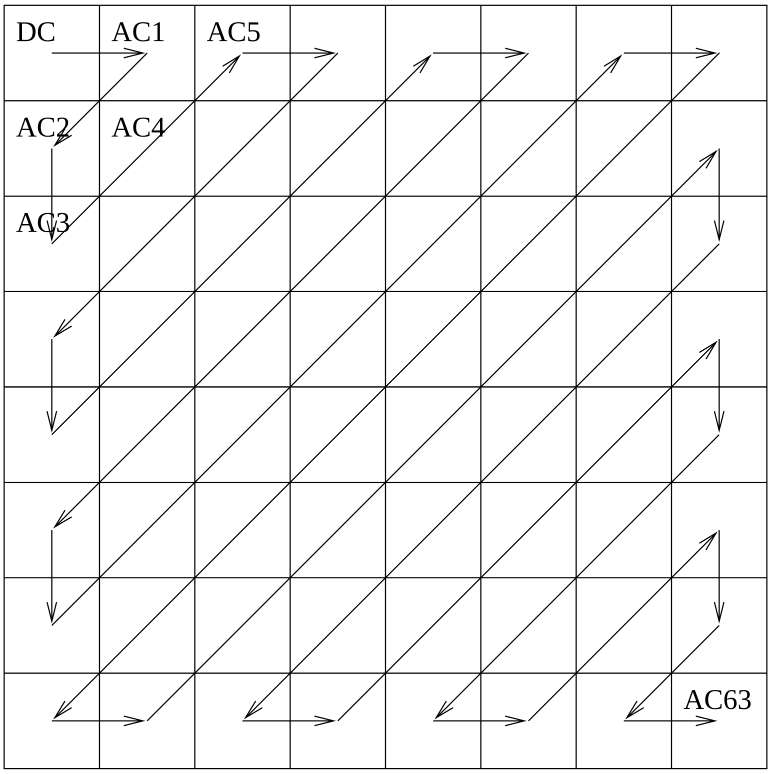

Figure 3.1: ZigZag ordering of DC and AC coefficients in an 8x8 block

Motion compensation will construct the well-known I-P-B frame patterns. In a P frame, each macroblock is encoded as a 16x16 area of past reference frame, plus a motion vector. Motion vectors are encoded, again, using the DCT, quantization, and run-length encoding. A macroblock may be skipped which is equivalent to a (0, 0) motion vector. B-frame is very similar to P-frame except that the motion vectors may also depend on future reference frame. The corresponding 16x16 macroblocks from past and future reference frames are averaged to form the B-macroblock.

The final MPEG stream coding structure looks as follow:

It starts with sequence header and ends with a sequence end code. In between, there is one or more group of pictures. Each group contains one or more pictures with types of I, P, and B. The picture is further divided into slices. Slice is the deepest level which has a header with 4-byte start code (see Figure 3.2). A slice consists of macroblocks, where each macroblock contains 4 luminance blocks followed by 2 chrominance blocks. The 4-byte start codes are defined for sequence, group, picture, and slice, to identify the starting position of different structural levels. It is important to point out that inside of one slice, there are only variable-length-codes chained in sequence and the correct interpretation is based on the prefix code property of Huffman coding (the codes in which no codeword is also a prefix of some other codeword are called prefix codes, see [19]). Missing or changing even one bit will change the meaning of the sequence. Obviously, the order of the code stream is also important for correct interpretation.

3.3 The Problems of Existing Schemes

We mentioned three different encryption algorithms for multimedia data, particularly for MPEG video streams. However, each of these schemes has its weakness in terms of either security level, or speed, or resulting stream size, or the combination of these metrics.

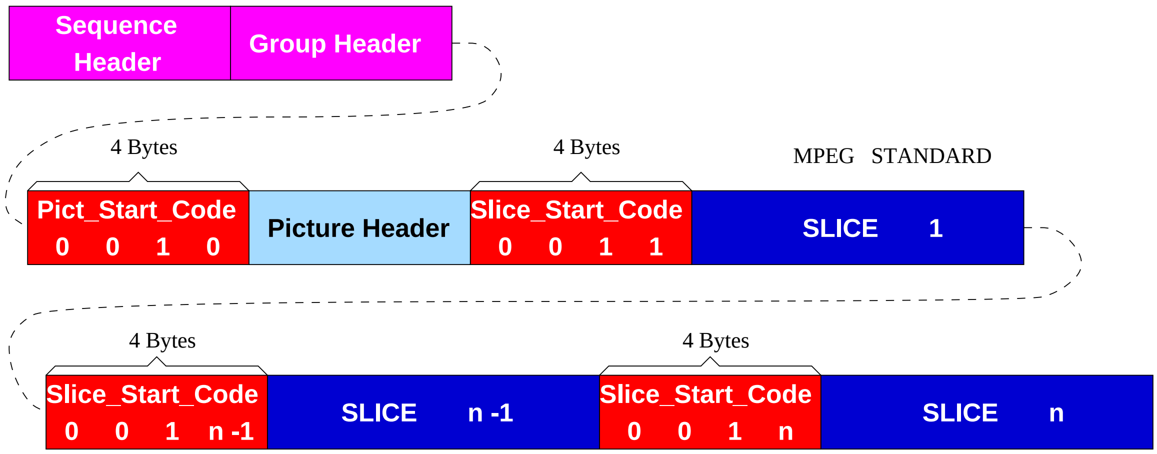

Figure 3.2: MPEG Video Layers

MPEG Video Clip Name	Frame Pattern I:P:B	Number of I-blocks		
		in I frames	in P frames	in B frames
bus.mpg	10:40:98	13200	11920	3744
space.mpg	647:0:0	207040	0	0
twister.mpg	206:206:824	247200	74400	76828

Table 3.1: Number of I-blocks in some MPEG video clips

For example, the concern about the Naive approach is the speed of processing due to the large size of MPEG files.

Agi and Gong have shown that great portions of the video are visible partly because of interframe correlation and mainly from unencrypted I-blocks in the P and B frames. Therefore, selective method of only encrypting I frames may not work [2].

For the Meyer and Gadegast's scheme, there are two problems. First, identifying the I-blocks in a P or B frame introduces overhead because one has to go through the MPEG stream bit-by-bit. Second, some MPEG streams only contain I frames. In this case, the selective algorithm is reduced to the naive algorithm. Even with the presence of P and B frames, the number of I-blocks in P or B frames could be of the same order as the number of I-blocks in I frames (see Table 3.1). This fact, coupled with the time-consuming process of identifying I-blocks, makes the naive algorithm a better choice in many cases.

There are also problems involved with ZigZag-Permutation Algorithm. Besides the lack of flexibility, this method has two fatal problems. First, changing the zig-zag order to a random order will result in image size increasing of about 25% to 60%, which is not tolerable. Second, the algorithm has serious security problems even with the added binary coin flipping sequence. Using the property of MPEG that the non-zero AC coefficients have the tendency to gather in the upper-left corner of the block, one can break this cipher easily. We will show the attacks in the following section.

3.4 Attacks on ZigZag-Permutation Algorithm

3.4.1 Known-plaintext Attack

As pointed out by the author L. Tang himself, the basic ZigZag-Permutation Algorithm can not withstand the known-plaintext attack. Therefore, he proposed to use the binary coin flipping sequence against known-plaintext attack. But this addition is useless. We will show how to break this technique:

Let us assume we know certain frame(s) of the video (the plaintext). By simple comparison between the plaintext and the corresponding encrypted frame(s), both key1 and key2 can be easily found (unless only one key was used for encrypting that frame which is highly unlikely because it is against the purpose of coin flipping sequence). We then use both key1 and key2 to decrypt the 1st block separately. Because the non-zero AC coefficients (named dct_coeff_next in MPEG standard) have the tendency to gather in the upper-left corner of the block, the one which has this tendency is the correct one and most likely the other one does not show this tendency because of the random permutation. We then apply the same strategy to the rest of the blocks. In this way, we only need about twice as much time as the normal MPEG decoding time to break it. Another security enhancement attempt of Tang's algorithm is to split each DC coefficient (named dct_coeff_first in MPEG standard) into two parts so that each part is a smaller number, group 8 DC coefficients together, encrypt them by standard encryption algorithm DES, and then apply permutation lists as described above. But, with the fail of the coin flipping sequence, this enhancement is essentially applying DES to DC coefficients and is a subset of SECMPEG level-2. Also, as Tang showed, if the DC coefficients are set to 0s without any permutation of AC coefficients, then the image is still comprehensible.

3.4.2 Ciphertext Only Attack

The basic ZigZag-Permutation Algorithm is also vulnerable to the ciphertext only attack. The attack is based on the fact that non-zero AC coefficients are gathered in the upper-left corner of the I-block.

We conduct some statistical tests which count the number of non-zero DC and AC coefficients from all blocks in an I frame. Two results, using two different MPEG video clips, are shown as follows:

```
                Number of non-zero coefficients in an I frame:

bus.mpg:

    Total number of blocks in one I frame is 1320. The number of non-zero
    DC coefficients in the tested frame is 1314, the number of non-zero
    $AC_1$ coefficients in the tested I frame is 1017, and so on.


               1314 1017  860  660  500  384  307  219
               1103  881  662  504  344  226  134   95
                917  704  513  356  249  140  101   52
                800  593  417  318  206  104   58   36
                734  472  374  247  139   79   23    6
                591  413  293  181   92   38    5    1
                544  347  246  130   50   12    1    0
                480  302  157   91   20    2    0 1314


space.mpg:

    Total number of blocks in one I frame is 320. The number of non-zero
    DC coefficients in the tested frame is 320, the number of non-zero
```

AC_1 coefficients in the tested I frame is 106, and so on.

320	106	48	28	12	25	24	17
202	70	22	6	5	5	1	0
134	42	16	7	4	1	1	0
134	36	15	8	6	1	1	0
88	17	12	3	3	0	0	0
62	26	8	3	2	0	0	0
44	22	5	6	1	0	0	0
47	16	5	2	0	0	0	320

From the tables above, we observe that

1) DC coefficients always have the highest frequency of non-zero occurrence, therefore, if we do the same analysis on a scrambled I frame, we can identify the position of DC immediately. Notice that both the upper-left (DC) and lower-right (AC_{63}) numbers are always same. This is due to the way how DC coefficient is split. We set all AC coefficients to 0 and use DC coefficient to create the image. Because we do not know the order of these two, we need to decode twice, one uses $DC + AC_{63}$ and the other uses $AC_{63} + DC$. Figure 3.3 shows the result of using only DC coefficients to decode.

2) the frequencies of non-zero AC_1 and AC_2 are among the top 6. For example, the frequency of the non-zero AC_1 in the I frame of bus.mpg is 1017 which is the third highest. The frequency of the non-zero AC_2 in the same I frame of bus.mpg is 1103 which is the second highest. The frequency of the non-zero AC_1 in space.mpg is 106 which is the 5th. Therefore, in order to decode the position of AC_1 and AC_2 coefficients, we need to try (6-1)x(6-2)=20 combinations (6-1 is because the position of the DC coefficient is known; 6-2 is because the position of the AC_1 coefficient is decided). Figure 3.4 shows the decoded frame using DC and AC_1, AC_2 coefficients.

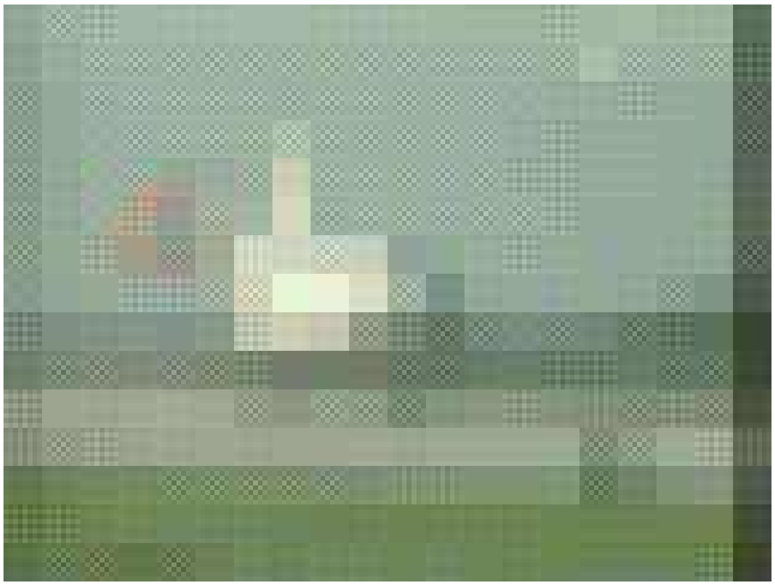

Figure 3.3: I frame decoded using only DC coefficients

3) the occurrence frequencies of non-zero coefficients AC_3 to AC_5 are among the top 10. To get correct AC_3, AC_4, and AC_5 coefficients, maximum of (10-3)x(10-4)x(10-5)=210 combinations need to be tried. Figure 3.5 shows the decoded frame using DC and first 5 AC coefficients. This is good enough to defeat the purpose of encryption in many circumstances. By comparison, we show the original frame in Figure 3.6.

3.4.3 Discussion

Based on the discussion above, we can conclude that the MPEG encryption method by replacing the ZigZag order with a random permutation list is NOT secure.

In general, we do not want to encrypt the information before compression process because the following statement holds: "The ciphertext produced by a good encryption algorithm will have (almost) no redundant information and is statistically indistinguishable from ran-

Figure 3.4: I frame decoded using DC, AC1, AC2 coefficients

dom numbers. No compression algorithm can further successfully compress random numbers." This statement by negation means that if a compression algorithm is applied to the resulting ciphertext of an encryption algorithm and can further reduce the ciphertext, then the encryption algorithm has security flaws. This is exactly what happened to the ZigZag-Permutation Algorithm.

3.5 Statistical Analysis of MPEG

Naive algorithm treats the MPEG stream just as the traditional textual data. The selective algorithms go one step further and take into consideration the MPEG structure. But it only uses the MPEG property of frame dependency. Obviously, MPEG stream is quite different from traditional textual data, therefore study of MPEG properties and its statistical behavior will lead to new directions of its security.

MPEG is a compression algorithm which removes redundant information from the image sequence. This means that it has a more uniform distribution of byte values and is therefore different from textual data. We are mainly interested in dealing with MPEG stream in *byte-by-byte* fashion for the following reasons: (1) it is easier to handle data byte-wise; and (2) the randomness is introduced at the byte level because of the variable-length Huffman codes used

Figure 3.5: I frame decoded using DC and first 5 AC coefficients

in the MPEG compression algorithm. Note that in dividing the MPEG bit stream into a byte stream, each unit is an integer between 0 and 255.

When studying the statistical behavior of MPEG streams, we consider MPEG video with different features, as shown in Table 3.2: for example, *bus.mpg* is a video with image resolution 352x240 pixels, MPEG frame pattern (I:P:B ratio) with 10:40:98, where on average I frame size is 13512 bytes, P frame size is 7657 bytes, and B frame size is 2723 bytes, and it is showing a moving bus with a rapidly changing background; on the other hand, *klara1.mpg* is a video of a lecturer during a class with little changing in the background, which translates into a very different frame pattern as well as different frame sizes in comparison to the previous video clip *bus.mpg*.

The measured metric in these streams is the *frequency of occurrence* of byte values. Figure 3.7 shows the frequency of MPEG byte values compared to the frequency of English letters. Figure 3.8 shows that by removing all headers in an MPEG stream, the frequency distribution is even smoother, e.g., the frequency of 0 is down from over 0.02 to less than 0.014. Figure 3.9 shows different streams, and we can see that all MPEG streams have a similar frequency distribution. Another interesting test is to analyze the statistical behavior of an MPEG file subset, e.g., the first or second half of the byte stream or a randomly chosen half of the byte stream (does not have to be consecutive). In these cases, the distribution does not change.

Figure 3.6: The original I frame

Figure 3.10 shows one such test. In fact, all tests on different MPEG streams show similar statistical results. In Table 3.3, we can see that the highest occurrence frequency is less than 0.0178 and the variance is in the magnitude of 10^{-6}.

Another important statistical behavior is the *frequency distribution of digrams*[1] among all 256x256 possible pairs. Our tests show that the occurrence of any of the 256x256 pairs is possible and valid. Our tests also show that the highest pair frequency is in the magnitude of 10^{-4} and higher frequency pairs vary from stream to stream. To measure the repetitiveness of the byte stream in MPEG video clips, the following steps are taken: (1) divide an I-frame into some equal length chunks, e.g., each chunk is 1/8 or 1/16 of the I-frame; (2) define random variable X as the number of occurrences of highest-frequency pair in one chunk; (3) calculate P($X > 1$); and (4) if a stream has a repeated pattern, then at least one Digram must be repeated, i.e., one pair appears more than once ($X > 1$). This behavior is demonstrated in Figure 3.11 within the interval [0, T]. However, for all tested video streams, the chance of repeating even the highest-frequency digram within a 1/16 chunk is less than 0.03. Table 3.4 summaries the Digram statistical analysis. We have also conducted tests considering two numbers that are not adjacent but have a fixed distance between them. We call this kind of pair the *Extended Digram*. Tests show that the Extended Digram has similar behavior as the Digram. The above

[1] pairs of two adjacent number

MPEG Video Name	Size	Pattern Frame I:P:B	Ave. Frame Length (bytes) of		
			I	P	B
bus.mpg	352x240	10:40:98	13512	7657	2723
flower.mpg	320x240	10:40:98	18936	10576	765
klara1.mpg	320x240	166:332:498	8775	5019	4359
space.mpg	160x128	647:0:0	1471	0	0
twister.mpg	320x240	206:206:824	6774	5677	4338
water.mpg	160x128	111:0:0	4246	0	0
bike.mpg	352x240	10:40:98	10153	6992	3326
car.mpg	320x240	86:43:172	11268	3425	518
coaster.mpg	288x192	80:40:160	8113	6666	2615
hula1.mpg	352x240	10:30:0	6620	2861	0
isochr.mpg	320x240	171:171:684	7471	3673	2774
orincsa.mpg	320x240	242:121:484	5313	1455	336
puffer.mpg	320x240	174:0:0	8039	0	0
simpson.mpg	192x144	171:171:342	4601	3662	1633

Table 3.2: Basic information of some MPEG streams

Name	highest frq.	variance	unicity
bus.mpg	0.01351 at 0	0.0000017323	24516
flower.mpg	0.01137 at 0	0.0000016178	23738
klara1.mpg	0.01004 at 0	0.0000009341	44326
space.mpg	0.01703 at 0	0.0000016991	28867
twister.mpg	0.01338 at 0	0.0000017311	24766
water.mpg	0.01268 at 0	0.0000018502	21171
bike.mpg	0.00657 at 0	0.0000004603	79587
car.mpg	0.01559 at 0	0.0000020275	22094
coaster.mpg	0.01529 at 0	0.0000022456	19382
hula1.mpg	0.00594 at 0	0.0000003110	116728
isochr.mpg	0.01020 at 0	0.0000008676	47651
orincsa.mpg	0.00690 at 0	0.0000005420	67017
puffer.mpg	0.01497 at 0	0.0000018027	24228
simpson.mpg	0.01780 at 0	0.0000030776	15353
English	0.14 at E	0.0011139678	32

Table 3.3: Statistical behavior of some MPEG streams

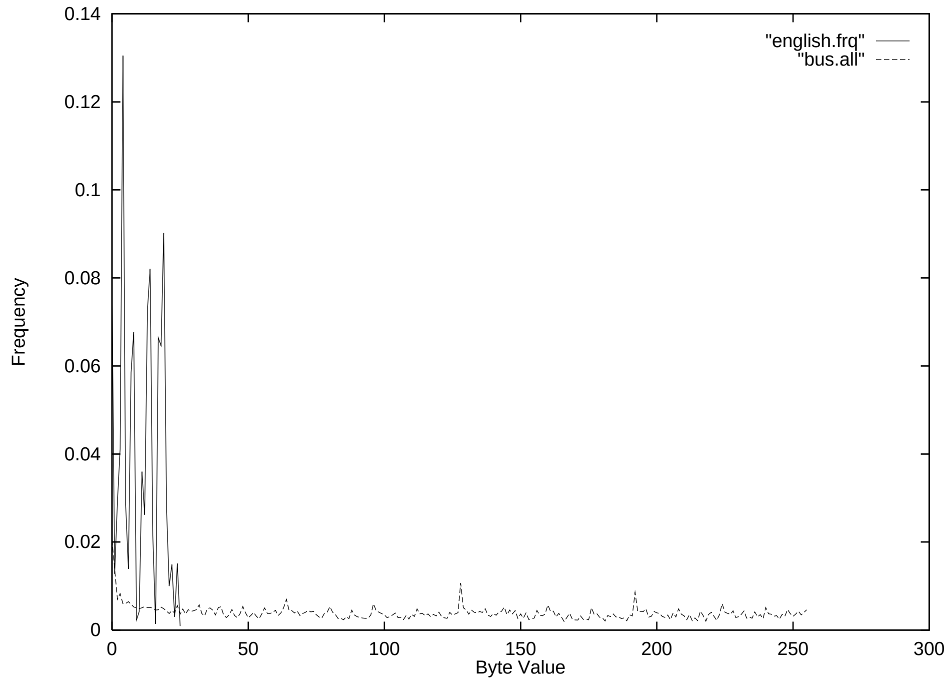

Figure 3.7: Frequency of Occurrence of Byte Values. (the distribution of an MPEG stream versus English)

discussed statistical analysis leads to the following assumption: **There is no repeated byte pattern within any 1/16 chunk of an I-frame.** This assumption will be utilized in the design of our VEA.

3.6 Video Encryption Algorithm - Basic Approach

In this section, we present the basic idea and approach of our Video Encryption Algorithm. It relies strongly on the discussion presented in Section 2.5. In order to achieve the highest security level, we consider the following:

1. Assume the chunk of an I-frame to be in the following form: $a_1a_2a_3a_4...a_{2n-1}a_{2n}$, where a_i is any byte value between 0 and 255.

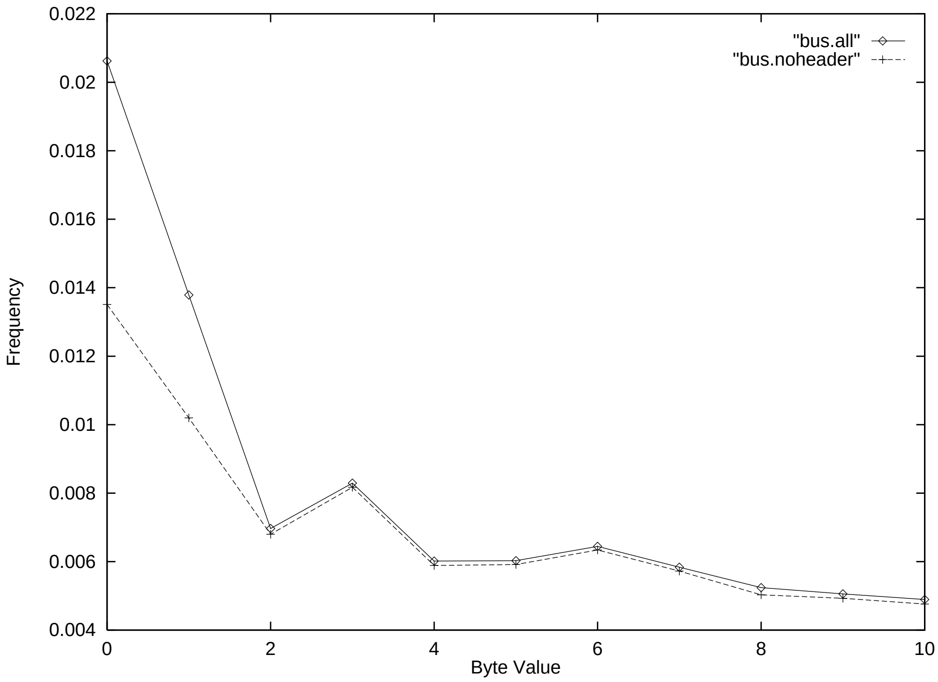

Figure 3.8: Frequency of Occurrence of Byte Values. (the distribution of an MPEG video with and without headers)

2. Choose odd-numbered bytes and even-numbered bytes to form two new byte streams. We call them *Odd List* and *Even List*, respectively.

3. Xor the two new streams, i.e.,

$$\begin{array}{r|cccc} & a_1 & a_3 & \ldots & a_{2n-1} \\ \text{Xor} & a_2 & a_4 & \ldots & a_{2n} \\ \hline & c_1 & c_2 & \ldots & c_n \end{array}$$

4. Choose an encryption function E to encrypt $a_2a_4...a_{2n}$. The resulting cipher-text has the form $c_1c_2...c_nE(a_2a_4...a_{2n})$.

It is easy to show that, if $a_2a_4...a_{2n}$ has *no repeated pattern*, then the secrecy depends on function E because $a_2a_4...a_{2n}$ is a one-time pad [54] which is well-known to be perfectly secure.

One benefit of our approach comes from the fact that we need 1 Xor to get $c_1c_2...c_n$ and 16 Xor's, using standard DES, to get $E(a_2a_4...a_{2n})$. In the case of Naive Algorithm, which applies standard DES to the whole stream, we need 2x16=32 Xor's. Therefore, the gain of our approach is:

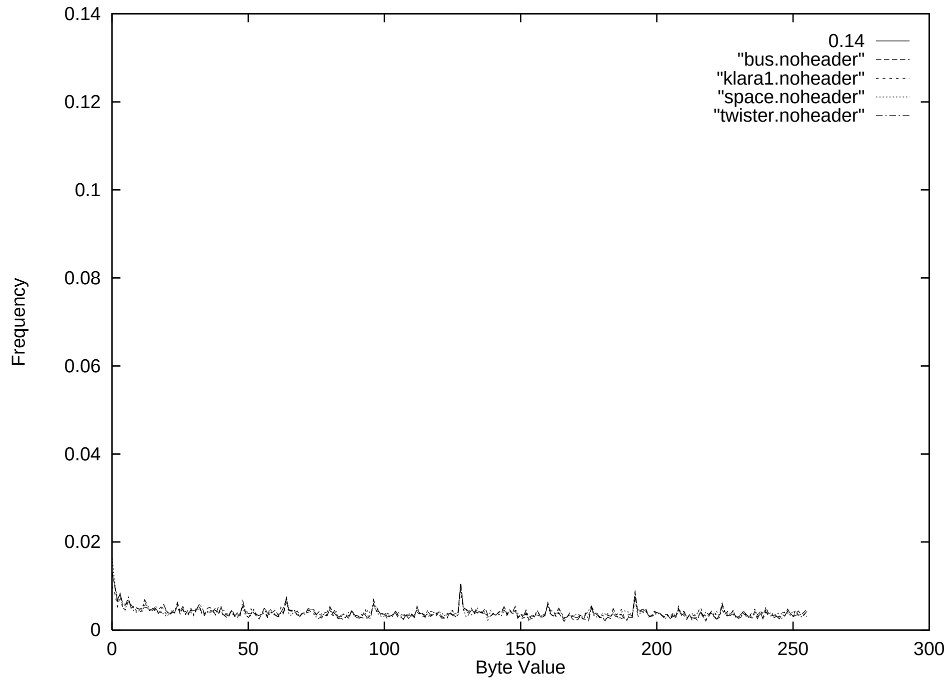

Figure 3.9: Frequency of Occurrence of Byte Values (the distribution of different MPEG streams). 0.14 is the frequency of English letter E.

$$(1 - \frac{1+16}{32}) \times 100\% = 47\%$$

3.7 VEA - Design and Implementation

In this section we present the integration of the basic method from Section 2.5 and key selection. This integration provides the fundamental basis of our *Video Encryption Algorithm*. Other issues such as space considerations, key distribution and experimental results are also presented.

3.7.1 Key Selection

$KeyM$: One concern regarding the method presented above is the regularity of choosing Odd and Even Lists, namely, if an attacker knows the plaintext of $a_1 a_3 \ldots a_{2n-1}$, then he or she can simply Xor them with $c_1 c_2 \ldots c_{2n}$ and get the other half easily. Although this seems very unlikely, we nevertheless need protection against this if the highest security level is to be achieved. We therefore introduce an 128-bit key which consists of randomly generated 0-1 bit sequences with an equal number of 0's and 1's (64 each). We call this key *KeyM*.

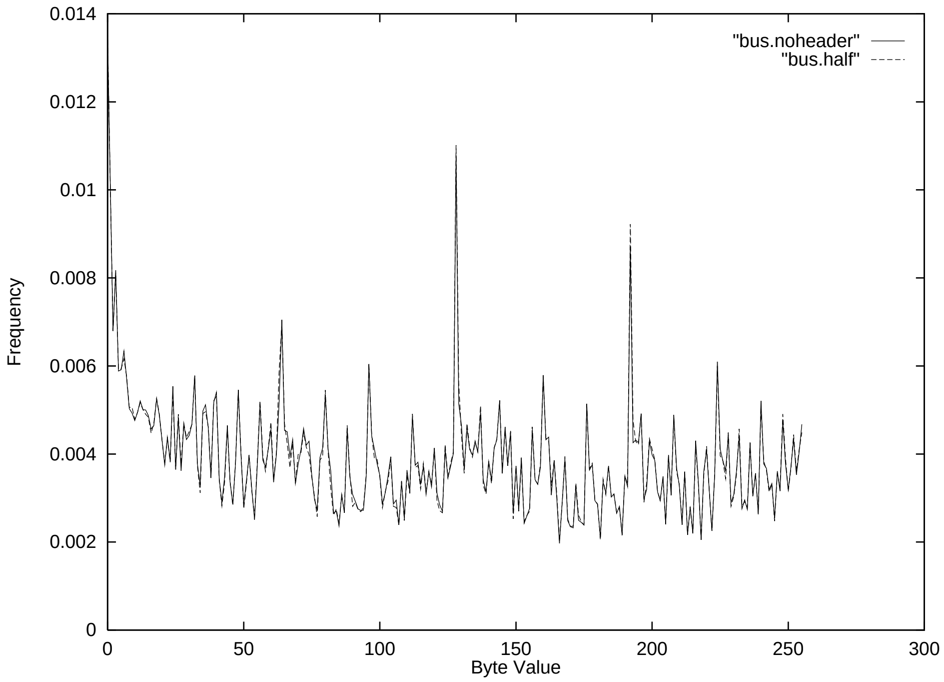

Figure 3.10: Frequency of Occurrence of Byte Values. (the distribution of a randomly chosen half of an MPEG stream)

The stream is divided into data segments of 128 bytes each. Two 64-byte sequences (i.e., Odd List and Even List) are generated from each 128-byte segment by repeatedly applying *KeyM* as follows: If bit_i of *KeyM* is 1 then put the i-th byte in Odd List, else put it in Even List.

Example: suppose $KeyM = 1\ 0\ 0\ 0\ 1\ 1\ \ldots$, then we have

$$
\begin{array}{cccccc}
 & a_1 & a_5 & a_6 & \ldots & \\
\text{Xor} & a_2 & a_3 & a_4 & \ldots & \\
\hline
 & c_1 & c_2 & c_3 & \ldots & c_{64}
\end{array}
$$

and the cipher-text is $c_1c_2...c_{64}E(a_2a_3a_4...)$. In doing so, the chance of correctly getting Odd List from plaintext (and so Even List) is less than 2^{-120}. In other words, the attacker needs to know all the plaintext. Furthermore, even if an attacker knows that the possible pair for c_1 is a_1 and a_2, for c_2 is a_5 and a_3, etc., then he still does not know which one belongs to the Odd List because there are 2^{64} possible outcomes.

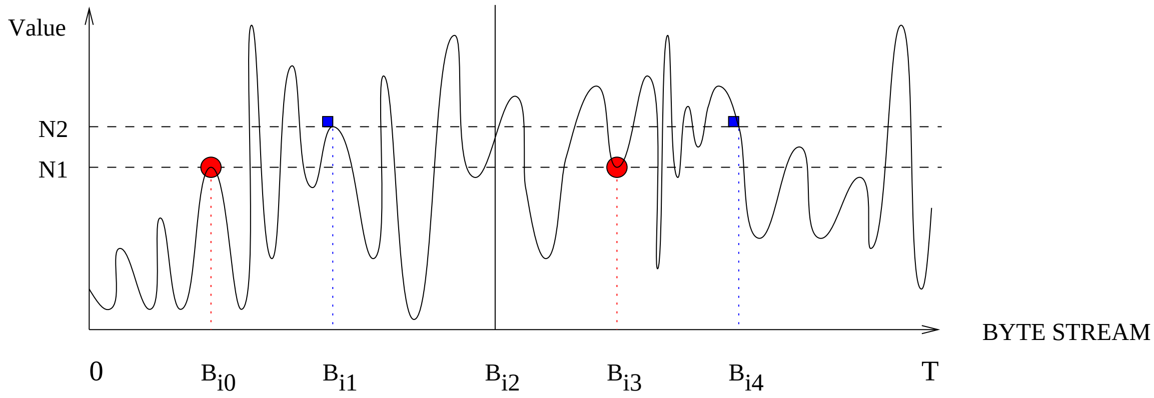

Figure 3.11: Repeated Pattern. If there is a repeated pattern, then at least one Extended Digram (N1 N2) can be found such that (N1 N2) appears at (B_{i0}, B_{i1}) and repeats at (B_{i3}, B_{i4}). $(|B_{i1} - B_{i0}| = |B_{i4} - B_{i3}|)$

Note that KeyM could also be a 256-bit random 0-1 sequence (128 0's and 128 1's) to satisfy even higher security requirements. Since video file sizes are typically several MBytes, we can afford to have such a long key.

Key_i : Another concern is the non-repeated pattern guarantee from Section 2.4. The statistical results show that the non-repeated patterns have a life time over only one 1/16 chunk. To get a non-repeated pattern with a length of 1/2 frame which equals to 8 chunks, each of the 8 chunks must be shuffled by different keys. We name these keys $Key_1, Key_2, ..., Key_8$. Each Key_i is a random permutation of (1, 2, ..., 32).[2] Each number is represented by at least 5 bits, therefore the length of Key_i is 32*5/8=20 bytes and the total length of $Key_1, Key_2, ..., Key_8$ is 160 bytes. We apply Key_1 to the first 32 bytes of Even List, then apply Key_2 to the next 32 bytes, and so on, and then repeat this process.

$KeyF$: Comparison of the MPEG frame size and the corresponding unicity suggests that the pattern of choosing Even List or Odd List should be changed for every frame. To achieve this goal, we assign each frame a 64-bit key and name it *KeyF*. *KeyF* contains a random permutation of (1, 2, ..., 16) with every 4 bits representing one number. We then apply *KeyF* to *KeyM* repeatedly and to the Key_i's to get new keys for each frame. Applying *KeyF* to

[2] In fact, 24 is sufficient because $24! \approx 2^{7}9$. This is considered unbreakable.

MPEG Video	Highest frq. pair	P(X>1) in a 1/8 chunk of I frame	P(X>1) in a 1/16 chunk of I frame
bus.mpg	0.000236 at (192, 3)	0.0612	0.0174
flower.mpg	0.000173 at (224, 3)	0.0641	0.0183
klara1.mpg	0.000198 at (6, 0)	0.0204	0.0055
space.mpg	0.000551 at (71, 3)	0.0048	0.0012
twister.mpg	0.000295 at (165, 41)	0.0265	0.0072
water.mpg	0.000349 at (3, 0)	0.0152	0.0040
bike.mpg	0.000214 at (250, 63)	0.0308	0.0084
car.mpg	0.000293 at (72, 139)	0.0650	0.0186
coaster.mpg	0.000522 at (74, 82)	0.0993	0.0294
hula1.mpg	0.000116 at (39, 9)	0.0043	0.0011
isochr.mpg	0.000246 at (9, 96)	0.0227	0.0061
orincsa.mpg	0.000391 at (165, 41)	0.0284	0.0077
puffer.mpg	0.000355 at (3, 0)	0.0213	0.0057
simpson.mpg	0.000381 at (3, 0)	0.0208	0.0056

Table 3.4: Digram Statistics.

KeyM seems redundant, but note that $KeyM \odot KeyF \odot Key_i \odot KeyF$ is not equivalent to $KeyM \odot Key_i \odot KeyF$ [3].

KeyE: We need a key for the encryption function E. We call this key *KeyE*. *KeyM*, Key_i (i=1..8), and *KeyF*'s can be encrypted using function E with *KeyE*. *KeyE* will be transferred using another secure channel or public key methods. We can also send *KeyM* and Key_i's using another secure channel or public key methods.

3.7.2 The Algorithm

Integrating the basic approach with the key selection, our algorithm consists of the following steps: (see also Figure 3.12)

1 For each MPEG frame, construct $HeaderBlock$.

2 Apply $KeyF$ to $KeyM$ and Key_i's.

3 For j-th 128-byte stream segment, compute i = (j mod 4).

[3] $key_a \odot key_b$ means apply key_b to key_a

4 Shuffle j-th 128-byte stream segment using $KeyM \odot KeyF$ and divide the resulting segment into four 32-byte parts sequentially. The first and third part serve as two Odd Lists and the second and forth one as two Even Lists.

5 Shuffle the first Even List using $Key_{2i+1} \odot KeyF$. The resulting Even List is: (1) XORed with the first Odd List giving us cipher-text c1 c2 ... c32; (2) Encrypted by Function E using $KeyE$ giving us cipher-text E1 E2 ... E32. Apply same steps to the other pair of Odd and Even Lists.

6 Repeat Step 3.

7 Repeat Step 1 for each Frame.

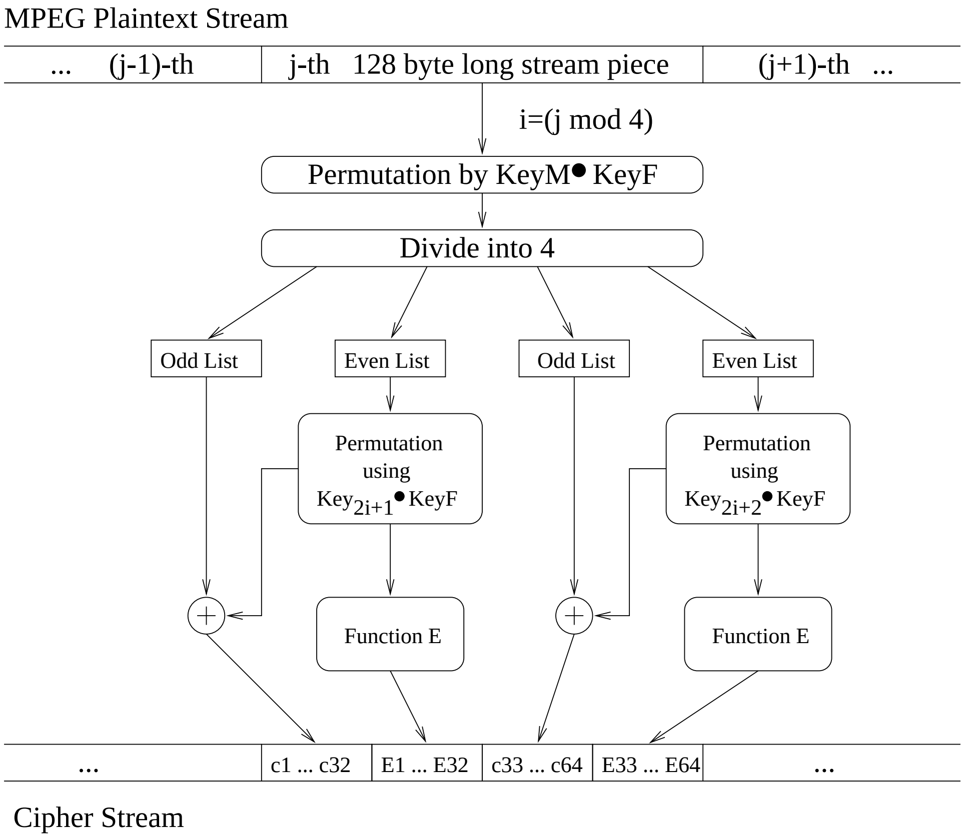

Figure 3.12: The Algorithm

3.7.3 Space Optimization

In a real-time Video-on Demand system, the MPEG stream is usually indexed and transmitted frame by frame. Therefore, the 4-byte Picture_Start_Code is not necessary. Also notice that the 4-byte Slice_Start_Code is always 001X with X meaning the slice number in the frame, these bytes are not necessary, too. The upper part of Figure 3.13 shows the standard MPEG frame structure. We can use the space of the unnecessary bytes to carry the key *KeyF*.

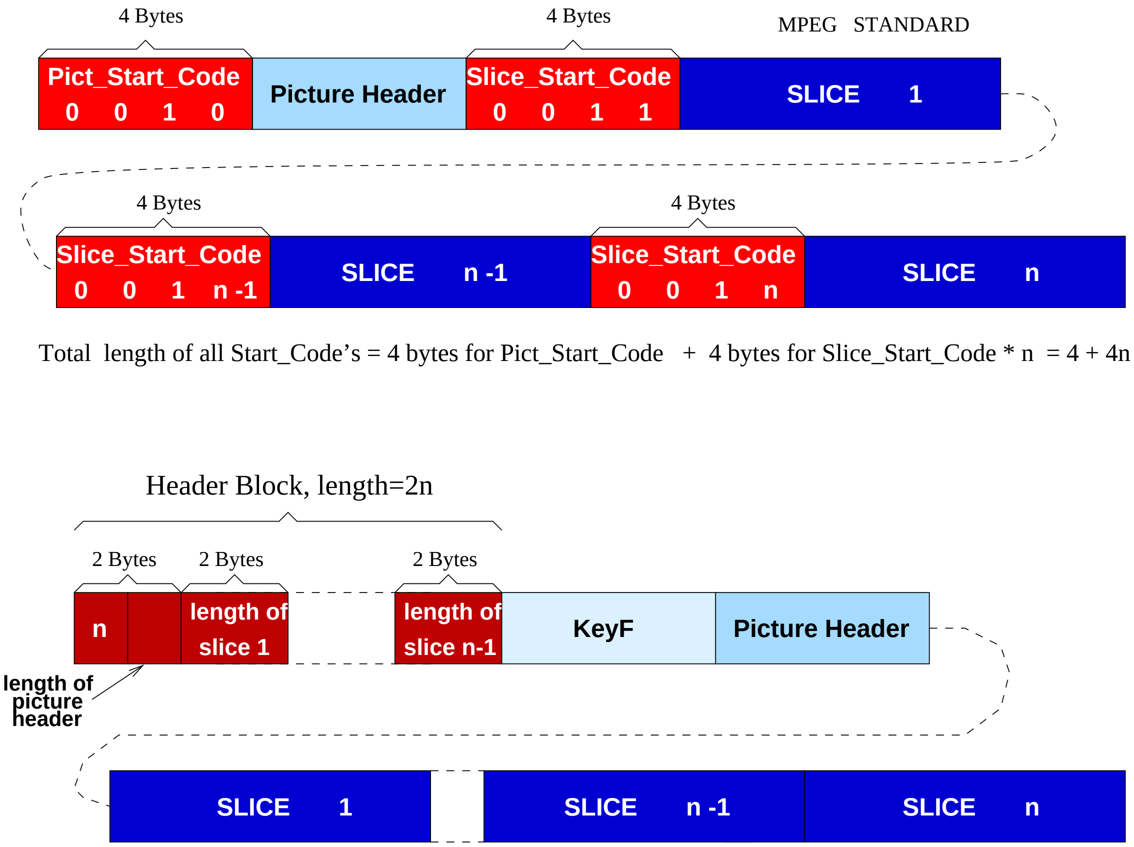

Figure 3.13: Header Block

We define for each frame a *Header Block* as shown in the lower part of Figure 3.13. The total length of the *Header Block* is $2n$ where n is the number of slices in one frame. The first byte is the total number of slices (n), the second byte is the offset from Picture_Start_Code to the first slice, and the following sequence of 2 bytes represents the length of each slice. The last two bytes represent the length of the $(n-1)$th slice. The *KeyF* then follows. When the destination receives the frame, it simply inserts 0010 (Picture_Start_Code) in front, reconstructs the Slice_Start_Codes, puts them into original position based on the slice length, and gets the

KeyF. There are several advantages of using the *Header Block* structure. First, when we encrypt *Header Block*, the slice structure is easily hidden. Second, the key is piggy-bagged and the total length is shorter in most cases. The total space cost for this strategy is $2n+8$ bytes with $2n$ bytes for the *Header Block* and 8 bytes for the key, compared to the original $4+4n$ with 4 bytes for Picture_Start_Code and $4n$ bytes for n Slice_Start_Code.

The *Header Block* and *KeyF* will be encrypted by *KeyE*. MPEG picture headers will not be encrypted because these headers keep regular information such as picture rate, picture size, picture type, quantizer scale, etc., and these information are useless without knowing payload data such as AC's and DC's.

3.7.4 Results

We implemented the above discussed algorithm and the experimental results confirm the analytically calculated gain of 47% in Section 2.5. Our results are shown in Table 3.5. We compared our results with the implementation of IDEA [4]. The gain is slightly larger than 47%, which could be explained by the overhead of IDEA implementation in PGP software. Furthermore, all tested streams show that the **encryption/decryption time per frame** is less than the frame rate time of 33.3 ms. This time-behavior provides very valuable feature for fast retrieval/playback service in telemedicine, newscasting and other security and time sensitive VOD applications.

3.8 Conclusion

In this chapter we presented two important research results: (1) the statistical behavior of MPEG video streams; and (2) the *Video Encryption Algorithm* for MPEG compressed video. VEA, a symmetric cryptosystem, utilizes the results of the statistical analysis of compressed video frames and provides very efficient, fast and secure encryption. The experimental results on various MPEG video streams verify our goal of providing a fast encryption algorithm, which also means that VEA can be part of the retrieval and playback process in Video-on-Demand applications.

[4] International Data Encryption Algorithm [35]

MPEG Video Streams	Total Time VEA	Per Frame VEA	Gain of VEA vs. IDEA	Total Time IDEA	Per Frame IDEA
bus.mpg	3.9779s	26.88ms	48.07%	7.6609s	51.76ms
flower.mpg	3.8208s	25.82ms	47.40%	7.2645s	49.08ms
klara1.mpg	29.7195s	29.84ms	48.65%	57.8817s	58.11ms
space.mpg	5.0372s	7.79ms	47.56%	9.6061s	14.85ms
twister.mpg	38.3258s	31.01ms	47.26%	72.6637s	58.79ms

Table 3.5: Experimental Results. Encryption (Decryption) Time

Chapter 4

Video Watermarking

4.1 Introduction

Digital watermarking techniques have been proposed in recent years as methods to protect the copyright of multimedia data. There are various watermarking schemes applied to images and several methods applied to audio and video streams. Among them, a large class of the watermarking schemes addresses invisible watermarks. The research and scheme design mainly focus on the issues such as *Perceptual Effects* – the watermark should be invisible and the distortion should be minimized; *Robust Design* – the watermark must be difficult to be removed and immune to multimedia data operations such as digital-to-analog and analog-to-digital conversion, requantization, dithering, resampling, rotation, scaling, cropping, etc. The watermark schemes must also be robust against *multiple-document attacks* [1].

In general, a watermarking scheme consists of the following processes:

1. Watermark Construction: generating the watermark.
2. Watermarking Process: embeding the watermark into the original multimedia data.
3. Verification Process: extracting the watermark and identifying the rightful ownership.

However, there are still open problems. One of them, as pointed out by Craver *et al* [23], is how to resolve the ***rightful ownership of the invisible watermarking schemes***. Craver *et al*

[1] Multiple-document attack uses t watermarked copies $D'_1, ...D'_t$ of the original document D to produce an unwatermarked document D^*.

attacked existing watermarking techniques by providing counterfeit watermarking schemes that can be performed on a watermarked image to allow multiple claims of ownerships. We refer to their attack as the *CMYY attack*. The rightful ownership problem is either not addressed at all or not addressed properly within current existing watermarking techniques. Furthermore, Craver *et al* defined the concept of *non-invertibility*. Informally, non-invertibility means that it is computationally impossible for an attacker to find a pair of a faked image and a watermark such that the pair can result in the same watermarked image created by the real owner. Craver *et al* also tried to design a non-invertible scheme which is a modified version of the watermarking method proposed by Cox *et al*. Unfortunately, their scheme cannot be proven to be non-invertible.

In this chapter, we present watermarking methods which resolve the rightful ownership and are successful against the CMYY attack. We introduce requirements on the watermark construction algorithm which include (1) using a standard encryption function such as DES [46], and (2) using the original image/video or other known bit stream information during the construction process. Because of these requirements, the non-invertibility of our proposed scheme is easily proved. Although our methods are mainly applied to MPEG-encoded video streams, they can be easily extended to many image schemes or other video schemes. We apply our scheme to raw images and prove its non-invertibility property as well.

4.2 Rightful Ownership Problem

The purpose of a watermark is to protect the owner's copyright. But without a careful scheme design and proper requirements on the watermark, an attacker can easily confuse everyone by manipulating the watermarked video (image, audio) and claim that he/she also is the original owner. This is called the "**rightful ownership**" problem.

Craver *et al* provided the following scenario: given a watermarked video (image), it is possible for an attacker to watermark the watermarked video (image) again using any watermarking scheme. This twice watermarked video (image) has both original and attacker's watermarks on it. Both the original owner and the attacker can claim the ownership, therefore, defeat the purpose of using watermark.

Using the original video clip (image) in the verification process can prevent the multiple ownership problems in some cases. However, even with the presence of the original video clip (image), the rightful ownership problem still exists. Craver *et al* showed that the following scenario is possible:

Assume that the original video clip (image) is V. The owner of the video clip uses watermark W to create a watermarked video clip (image) V_W, and publishes V_W. We denote this process as

$$V \oplus W => V_W \tag{4.1}$$

The attacker creates a watermark W_F, without knowing V, extracts W_F from V_W and creates a counterfeit video clip (image) V_F. Notice that,

$$V_F \oplus W_F => V_W \tag{4.2}$$

In this way, the attacker can use V_F as his "original" and claim the ownership of V_W.

If (2) can be achieved, then that scheme is called *invertible* watermarking. Otherwise, it is called *non-invertible* watermarking.

Now we formally define the invertibility and non-invertibility properties of a watermark scheme. Table 4.1 shows the basic notations of all variables used in this paper.

DEFINITION 1 [Weak Non-Invertibility]

A watermarking scheme *(E, D, C)* is (strongly[2]) invertible if, for any video (image) V_W, there exists a mapping E^{-1} such that

$$E^{-1}(V_W) = (V_F, W_F) \text{ and } E(V_F, W_F) = V_W$$

where the construction of E^{-1} is computationally feasible and W_F belongs to the set of allowable watermarks, V_W and V_F are perceptually similar, and, $C(D(V_W, V_F), W_F, \delta) = 1$.

[2] Craver *et al*'s original definition did not use this terminology.

Notation	Meaning
V	Original video clips (or image)
V_W	Watermarked video clips (or image)
W	Owner's watermark
W'	Extracted (recovered) watermark
V_F	Falsified original video clips (or image)
W_F	Attacker's watermark
E	Watermark inserting procedure: $E(V, W) = V_W$
D	Watermark extracting procedure: $D(V_W, V) = W'$; if V is not needed: $D(V_W) = W'$
C	Similarity testing: given threshold δ, X and Y are similar iff $C(X, Y, \delta) = \{ \begin{array}{l} 1, \text{ if } c \geq \delta; \\ 0, \text{ otherwise.} \end{array}$ where $c = cor(X, Y)$ is the correlation function of X and Y, X and Y are either both images, or both watermarks.
x_i	i-th element of X, where X could be V, V_W, W, etc.
α	Scaling factor

Table 4.1: *Basic Notation of Variables.*

Otherwise, *(E, D, C)* is (weakly[3]) non-invertible. □

Definition 1 comes from the Craver's paper [23]. However, the requirement of $E(V_F, W_F) = V_W$ is too restrictive (too strong) on the invertibility because V_W could be altered due to the noise and even the original V and W may not satisfy $E(V, W) = V_W$ in the verification process. On the other hand, the requirement of $E(V_F, W_F) = V_W$ is too loose (too weak) on the non-invertibility. We replace this requirement with $E(V_F, W_F) = V_W'$ where $C(V_W', V_W, \delta) = 1$.

DEFINITION 2 [Strong Non-Invertibility]

A watermarking scheme *(E, D, C)* is weakly invertible if, for any video (image) V_W, there exists a mapping E^{-1} such that

$$E^{-1}(V_W) = (V_F, W_F) \text{ and } E(V_F, W_F) = V_W'$$

[3] Craver *et al*'s original definition did not use this terminology.

where the construction of E^{-1} is computationally feasible and W_F belongs to the set of allowable watermarks, $V_W^{'}$ and V_F are perceptually similar, i.e., $C(V_W^{'}, V_F, \delta_V) = 1$, $V_W^{'}$ and V_W are perceptually similar, i.e., $C(V_W^{'}, V_W, \delta_V) = 1$, and extracted watermark $D(V_W^{'}, V_F)$ is similar to W_F, i.e., $C(D(V_W^{'}, V_F), W_F, \delta_W) = 1$.

Otherwise, *(E, D, C)* is strongly non-invertible. □

Currently, there are two proposals to solve the ownership problem. First solution is presented by Craver *et al.* They suggested that one should use a one-way function to map 1000 bits of the original image V to a bit sequence b_i (i=1, 1000) and then use the bit sequence to arbitrate between two different marking functions, i.e., if $b_i = 0$ then use formula $v_i(1 + \alpha w_i)$, and if $b_i = 1$ then use $v_i(1 - \alpha w_i)$, where v_i is the i-th element of the 1000 highest AC coefficients of the original image and w_i is the i-th bit of the watermark. It is believed (although it has not been proved) that this scheme is secure against any trial and error type of attacks that an attacker can attempt in order to construct a falsified original.

There is a very important assumption (regulation) in the above algorithm which is: w_i must be positive. In general, we need to enforce that the construction of any watermark has to follow certain rules. Otherwise, the ownership problem can not be solved. Craver *et al* gave one such an example which shows that if w_i could also be negative in the above scheme, then the scheme is invertible. Here we show the importance of the watermark construction requirements by another example which is an extreme case.

EXAMPLE 1 [An Extreme Case] If there is no restriction on the construction and the format of the watermark, an intruder can simply choose one bit v_{Wi} from V_W, set $v_{Wi} = v_{Wi} - 1$ to form his falsified original V_F and claim his watermark as (0 ... 0 1 0 ... 0) where only i-th bit is 1. It is easy to see that this simple scheme satisfies the invertibility definition 2. □

There is a class of invisible watermarking schemes which do not use original image (video clips) in the verification process. We will refer to these schemes as the *Self-Proof Class*. Because the use of the Self-Proof Class is simpler than that of other watermarking schemes, it is interesting to study its non-invertibility. We want to know "Are there any non-invertible schemes which belong to the Self-Proof Class?". Unfortunately, the answer is NO.

THEOREM 1 The Self-Proof Class is inherently invertible.

PROOF:

Let (E, D, C) be a scheme from the Self-Proof Class. Because this scheme uses only the watermarked image (video clip) V_W in the verification process, there is no way to verify how the watermark is created and how the original image(video clip) looks like. Because there is no restriction on the structure of the watermark and the original, any kind of watermarks and originals are legitimate. Therefore, an attacker can always use EXAMPLE 1 described above to make his/her ownership claim. This means any scheme of Self-Proof Class is invertible. □

Of course, one can add a requirement to the watermark construction in the Self-Proof Class schemes. But in order to verify the watermark construction, something other than the watermarked image must be presented. This contradicts the concept and the definition of the Self-Proof Class scheme which requires only the watermarked image during the verification process.

Another attempt to resolve the ownership problem is provided by Wolfgang and Delp [67]. Their method uses timestamps to generate the watermark. The owner with the earliest timestamp is the true owner. However, this scheme can be easily defeated because timestamps can be manipulated. For example, the time of an event, such as "the Berlin Wall came down", is well-known. If an attacker uses that time as his/her timestamp in watermarking the video/image taken during that event, then who is the original owner? Another case can occur when the real time clock is displayed somewhere in the picture, such as in the videos of news broadcasting, basketball games, etc. In these cases, the timestamps can be accurately determined by attackers according to the real clock. Again, if an attacker uses these timestamps, then who is the original owner? Based on the above discussion, we conclude that the use of timestamp is not a good way to solve the watermark ownership problem.

4.3 Non-Invertible Watermarking Schemes

One possible approach against CMYY attack is to make a strict requirement on the construction of the watermark and bind the watermark to the original video/image itself or some publicly known number sequence such as π. This will greatly limit the choices for the watermark W_F and falsified "original" V_F of an attacker. Clearly, if it is computationally impossible for an attacker to find both W_F and V_F which satisfy formula (2), then the non-invertibility is achieved. In this section, we will first derive such a scheme which applies to the MPEG-encoded video streams and then derive another one for a raw image or an uncompressed video stream. We will prove the non-invertibilities of both schemes.

4.3.1 MPEG Based Non-Invertible Watermarking Scheme

In this subsection, we will refer to V as a single I frame within the I, P, B sequence of a MPEG stream.

4.3.1.1 Watermark Construction Process

We concentrate on the description of the watermark construction applied to a single image V encoded in JPEG format. This description can be easily extended to the I frames of MPEG video clips. Our method utilizes ideas from direct sequence spread spectrum communications. The watermark construction algorithm consists of several steps:

First, we choose a standard encryption function, such as DES, and a key KEY. In Zig-Zag order, we scan each block of the DCT-transformed image.

Let $AC_{b,l}$ denote the value of the l-th AC coefficient in the block b, where $b = 1 \ldots nb$, and nb is the number of blocks in V, $l = 1 \ldots 63$. Let

$$NZ_{b,l} = \begin{cases} 1 & \text{if } AC_{b,l} \neq 0, \\ 0 & otherwise. \end{cases} \tag{4.3}$$

Let $nAC_l, l = 1...63$ denote the total number of non-zero AC coefficients at the l-th position of each block, i.e.,

$$nAC_l = \sum_{1 \le b \le nb} NZ_{b,l} \tag{4.4}$$

For example, if $AC_{1,5} = 1$, $AC_{2,5} = -2$, $AC_{1001,5} = 2$, $AC_{b,5} = 0$ in all other blocks, then $nAC_5 = 3$.

Second, we transform the first m $(1 \le m \le nb)$ nAC_l into binary numbers and concatenate those binary numbers to form a PAD, i.e., $PAD = nAC_1...nAC_m$. The choice of m will depend on the number of bits which can be embedded into the frame V. For example, for a standard MPEG-1 video with picture size 352x288 (pixels), there are overall 1584 blocks. Therefore, 11 bits are required to represent each nAC_l because $2^{10} = 1024 < 1584 < 2048 = 2^{11}$. If we are allowed to embed a total of 101 bits, m will be $\lceil 101/11 \rceil = 10$.

Third, we apply DES with KEY to PAD and get $EPAD = DES_{KEY}(PAD)$.

Forth, let us consider definitions of the following bit sequences:

- Let A_j be a bit sequence, where j=1 ... total number of bits allowed to embedded in V, such that

$$A_j = \begin{cases} -1 & \text{if } EPAD_j = 0, \\ 1 & \text{otherwise.} \end{cases} \tag{4.5}$$

- Let B_i be a bit sequence, such that

$$B_i = A_j, \qquad (j \times C_r) \le i < ((j+1) \times C_r) \tag{4.6}$$

where C_r is the chip-rate.

- Let p_i be a bit sequence, such that

$$p_i = \begin{cases} -1 & \text{if i-th bit of } DES_{KEY}(V) \text{ is } 0, \\ 1 & \text{otherwise.} \end{cases} \tag{4.7}$$

Last, we construct the watermark bit sequence by combining the sequences B_i and p_i:

$$w_i = \alpha \times B_i \times p_i, \mathrm{i} = 1 \ldots \text{ (Width x Height)} \tag{4.8}$$

We will discuss how we select the scaling number α in Subsection 4.1.2.

The Figure 4.1 shows a graphical representation of the whole watermark construction process. It is applied to the standard MPEG-1 frame with a size of 352(Width)x288(Height) pixels.

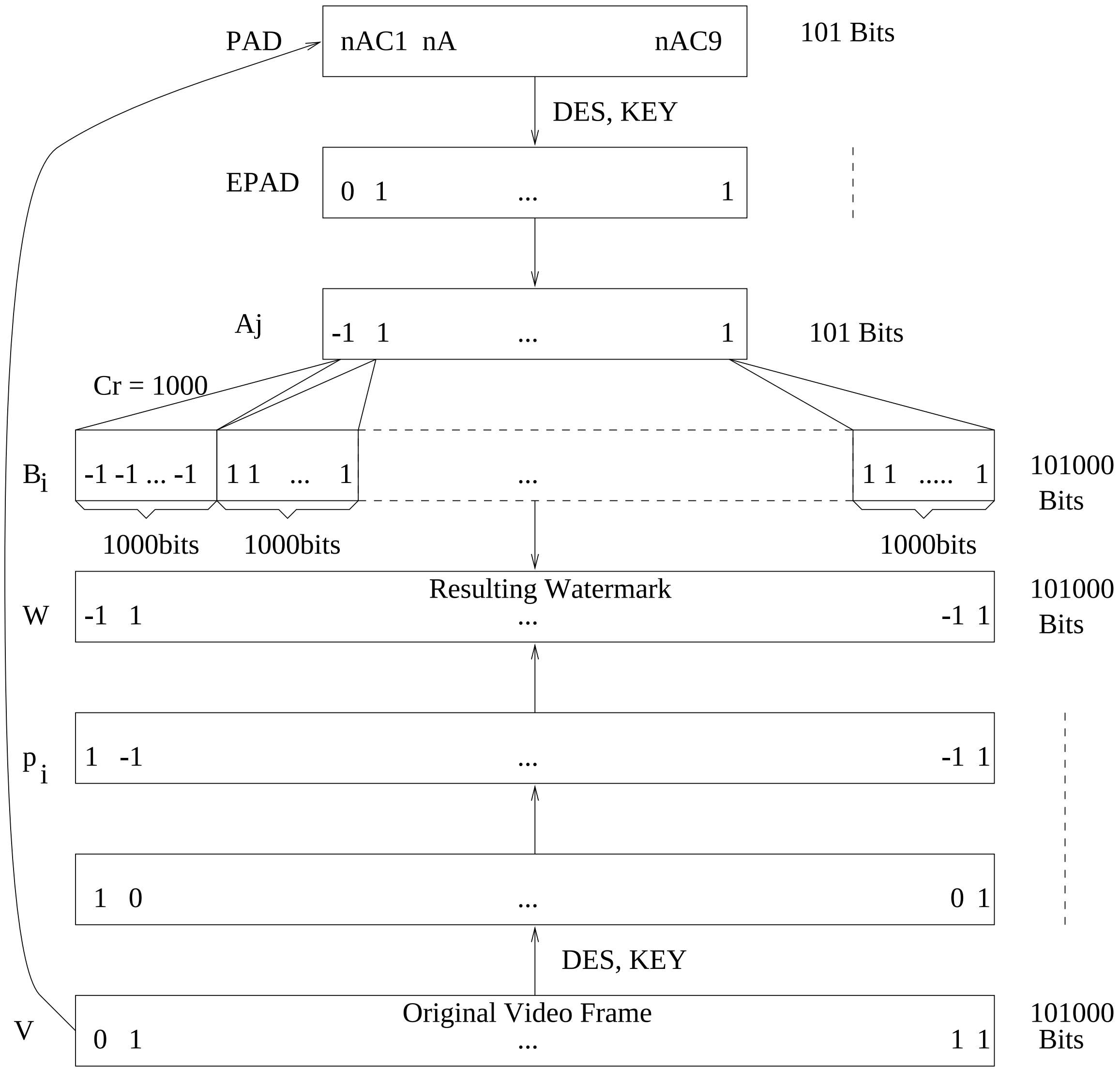

Figure 4.1: Construction of the watermark.

4.3.1.2 Watermarking Process

There are several assumptions and requirements which need to be mentioned before we describe our scheme.

A1 We derive a watermarking scheme for MPEG-compressed video without fully decoding the video to raw image sequence and again encoding. We decode the video up to the DCT coefficients and then embed a watermark into the stream at the DC and AC coefficient level.

A2 We set $\alpha = 1$. Tests on various MPEG clips show that, with chip-rate [4] 1000, if we watermark the DC coefficients, then the watermarked video has a lower quality and the watermark is visible (see Figure 4.2). This is a violation of the main requirement on watermarking - perceptual effects of watermarking should be invisible. One way to solve this problem is to increase the chip-rate, for example, to 5000. This, however, will decrease the amount of information that can be embedded by a factor of 5. Therefore, such an approach is not a good solution. We choose not to mark any DC coefficients. (Setting α to a small number can also avoid the quality degradation, but this means that there are many choices for α in one scheme and different α can be chosen in different schemes. This uncertainty gives an attacker freedom to manipulate the verification process. Therefore, we fix $\alpha = 1$.)

A3 We only watermark I frames since they are the most significant frames in a video stream. The watermark of I frames will be carried over to P and B frames due to the dependency between I, P, and B frames. It is true that an attacker could drop all watermarked I frames to get an unwatermarked video stream. However, although P and B frames may contain some I-blocks which can be decoded by themselves, there is enough dependency between the I frame and the P and B frames, therefore, the quality of the P or B frames will be bad enough if the I frames are dropped.

A4 We want to achieve that the total length of a watermarked stream is less than or equal to the total length of the original stream, so that the watermarking process does not defeat the compressing process of MPEG.

[4] Chip-rate: In spread spectrum systems, this is the rate at which the discrete signal is applied.

Figure 4.2: A frame from the video clip "Hula". Both DC and AC coefficients are watermarked ($\alpha = 1$).

A5 In our scheme, the original video stream and a key are needed for both the watermark creation and the verification process. This is justified by THEOREM 1.

With the above assumptions and requirements, we design an algorithm as shown in Figure 4.3 called ALGORITHM 1.

The main steps of the ALGORITHM 1 are:

M1 A new watermark is created based on the encrypting information from the original I frame V; We denote the process as $w_i = DES_{KEY}(v_i)$. We then put the sequence w_i into a two dimensional matrix of (Height x Width). The result is a watermark in the form of a raw image.

M2 DCT is applied to the watermark;

M3 An AC coefficient in the original image V is marked if the length of the resulting VLC (Variable Length Code) does not increase.

Let nAC_l denote the number of l-th AC coefficients in zigzag order in the I frame, where $l = 1, ..., 63$.
Let chip-rate be C_r, then $e = \lfloor \frac{Height \times Width}{C_r} \rfloor$ bits can be embedded in a frame with size (Height x Width).
Let each nAC_l be presented by d bits, $d = \lceil log_2(\frac{Height \times Width}{8 \times 8}) \rceil$.
Let WAC_l denote the l-th AC coefficient of any block in the watermark W.
Let PAD $= nAC_1$ ... nAC_m, where $m = \lceil \frac{e}{d} \rceil$.

For each I frame
{ /* Construct the watermark */
    Choose a KEY and a standard encryption function such as DES;
    $EPAD = DES_{KEY}(PAD)$;
    Let A_j = -1 if $EPAD_j$ is 0, otherwise 1;
    Let $B_i = A_j, (j \times C_r) \leq i < ((j+1) \times C_r)$;
    Let $p_i = -1$ if i-th bit of $DES_{KEY}(Iframe)$ is 0, 1 otherwise;
    The watermark $w_i = \alpha \times B_i \times p_i, i = 1...(Height \times Width)$;
    Put w_i into $Height \times Width$ matrix and then divided into $\frac{Height \times Width}{8 \times 8}$ 8x8 blocks;
    Do DCT on each blocks and get corresponding AC coefficients WAC_l;

    /* Watermarking process */
    For each block in I frame
        For each non-zero-AC_l
        {   add WAC_l to AC_l;
            if (Val(WAC_l + AC_l)==0)
                if (length(new code of next-non-zero-AC) <= length(next-non-zero-AC)+len(AC_l))
                    { mark AC_l, goto next block; }
                else
                    if (by removing the last non-zero-AC, the total length does not increase)
                        { mark AC_l, remove the last non-zero-AC, goto next block; }
                    else
                        { do not mark AC_l, goto AC_{l+1}; }
            else
                if the total length does not increase
                    { mark AC_l, goto AC_{l+1}; }
                else
                    if (by removing the last non-zero-AC, the total length does not increase)
                        { mark AC_l, remove the last non-zero-AC, goto next block; }
                    else
                        { do not mark AC_l, goto AC_{l+1}; }
        }
}

Figure 4.3: ALGORITHM 1

M4 An AC coefficient is marked and the last non-zero AC coefficient (in the zig-zag order) of a block in the original image V is dropped if the total length of the resulting VLC does not increase. No more than two AC's can be dropped within a block.[5]

We now discuss the results and the implications of the proposed scheme.

R1 Formula (nAC_l in V_W) $<$ (nAC_l in V) always holds. This is because, for each block, there are three possible outcomes: the number of non-zero AC's decreases by 0 (unchanged), 1, or 2 in comparison with the original video frame V. nAC_l in V_W depends only on the original V and KEY. It is easy to see that the total number of non-zero AC coefficients in V_W decreases. In addition, the values of some non-zero AC coefficients in V_W can also be changed. Such changes also depend only on V and KEY.

R2 The randomness of the embedded information is guaranteed by the encryption function DES. p_i in the spread spectrum scheme is the *Pseudo Random Noise Code*. By applying DES to V, we can create a Pseudo Random sequence of values 1/-1 which can be used as p_i. By choosing a different KEY, a different p_i sequence is created. The KEY is safe due to the DES encryption algorithm which means that it is computationally impossible for an attacker to find out the key even with the knowledge of V.

R3 The information contained in PAD is essentially the number of first m non-zero AC coefficients in the frame V. This information is unknown without the knowledge of the original V.

4.3.1.3 Verification Process

The verification process justifies the ownership claim and consists of three steps:

J1 The claimant is required to provide his/her original video clip V, the key KEY, and his/her watermark W. Then the trusted third party verifies the creation of the watermark. If the watermark is confirmed, then Step J2 is applied. Otherwise, the ownership is denied to the claimant.

[5] Two AC's can be dropped only when one becomes 0 and another, which must be the last non-zero AC coefficient, is dropped.

J2 The trusted third party applies ALGORITHM 1 described in Figure 4.3. Let the resulting watermarked video/image be V_v. As the next step, apply Step J3.

J3 The trusted third party compares the resulting V_v with the published watermarked video clip/image V_W. If V_v and V_W are similar, i.e. $C(V_v, V_W, \delta) = 1$, then ownership is granted to the claimant, otherwise ownership is denied.

4.3.1.4 Proof of Non-Invertibility for ALGORITHM 1

In this section, we will show that the watermarking algorithm presented in Section 3.4.1.2 together with the suggested watermark construction presented in Section 3.4.1.1 form a non-invertible watermarking scheme.

THEOREM 2 ALGORITHM 1 is non-invertible.

PROOF:

We need to analyze two major steps in the scheme: a) watermarking process and b) watermark generation.

a) The watermarking process in our scheme is fixed and the result depends exclusively on the original I frame V and the watermark W. If the watermark W is generated by DES as described in Section 3.4.1.1, then W is a random bit sequence. Obviously, applying our scheme to two different random bit sequences will result in two different watermarked video frames $V1$ and $V2$, such that $C(V1, V2, \delta) \neq 1$.

b) Clearly, the attacker can choose a bit sequence, i.e. choose a watermark, such that resulting V1 and V2 satisfy $C(V1, V2, \delta) = 1$. But from R2. and R3. above, we can see that the watermark in our scheme is fixed and depends only on the original I frame V and the key KEY. Because the attacker has to show that his watermark is the result of applying DES to his original, he/she has to find the key based on his/her chosen watermark. This is equivalent to "given ciphertext, find the plain-text and the key" or to break DES[6] which is impossible.

[6] Currently, Double-DES or Triple-DES, which use two or three different keys, can be considered as safe. Throughout this paper, DES means either Double-DES or Triple-DES.

In summary, we can conclude that ALGORITHM 1 is non-invertible. □

From the second half of the proof above, we can see that, given a watermark, it is impossible to find the corresponding original because this is also equivalent to "given ciphertext, find the plain-text and the key" or to break DES. Therefore, we have the following implication:

COROLLARY 1 If the original video/image is known but the key is unknown, it is still impossible for anyone to create the same watermarked video clips (image) using ALGORITHM 1.

4.3.1.5 Discussion

COROLLARY 1 implies that the construction of the watermark plays an essential role in our scheme to solve the rightful ownership problem. By using the encryption function as a "black-box", the watermark can not be arbitrarily chosen by an attacker. Therefore, checking the watermark construction is needed during the watermark verification process.

COROLLARY 1 seems to indicate that the original video/image information is not needed during the verification process. But this is not true. Without the original, we would end up with a scheme of the Self-Proof Class which would be an invertible one based on THEOREM 2. However, note that the "original" does not have to be a true image or part of the video clip. It could be some publicly known number sequence. For example, we can Xor the original V with the first 100000 digits of number π as the "original" for generating the watermark. In this case, the creation of p_i in ALGORITHM 1 should be modified as:

$$p_i = -1 \text{ if i-th bit of } DES_{KEY}(\pi\ Xor\ V) \text{ is 0, 1 otherwise.}$$

We also notice that $EPAD$ is actually redundant in the watermark construction process. We can simplify the process and use the random sequence p_i as the watermark as long as p_i is created by applying DES (or other encryption functions).

Finally, there is a concern that ALGORITHM 1 may be subject to the *multiple-document attack*: the AC coefficients which were dropped in one watermarked version may appear in other versions which were created by applying different keys; the AC coefficients which were modified in one watermarked version might be untouched in other versions. Therefore it is possible for the attacker to guess the original value of the AC coefficients if there are enough versions available. One way to solve this problem is to use a master key to preprocess the original video clip (image) by applying the proposed watermarking algorithm and then always use this preprocessed version as the "original". In doing so, the original information, which was changed by the preprocessing, will never be exposed.

4.3.2 Raw-Image-Based Non-Invertible Watermarking Scheme

By applying the similar idea discussed in Section 3.4.1, we can give a modified version of the scheme described by Cox *et al* [21]:

ALGORITHM 2 Given an original image V, let us choose a key KEY. Then we create the watermark $W = (w_i)$ by applying a standard encryption function such as DES, i.e.,

$w_i = DES_{KEY}(v_i)$, where v_i is the i-th pixel of V.

Watermarked V_W is constructed as:

$$v_{Wi} = v_i \times (1 + \alpha \times w_i) \tag{4.9}$$

THEOREM 3 ALGORITHM 2 is non-invertible.

PROOF: (by contradiction)

First, let us prove the non-invertibility of ALGORITHM 2 under DEFINITION 1.

Let us assume that ALGORITHM 2 is invertible. Then by DEFINITION 1 we can find v_{Fi} and w_{Fi} in reasonable time, such that

$$v_{Wi} = v_{Fi} \times (1 + \alpha \times w_{Fi}) \tag{4.10}$$

On the other hand, ALGORITHM 2 requires that

$$DES_{KEY_F}(v_{Fi}) = w_{Fi} \tag{4.11}$$

We combine these two formulas, the requirement of ALGORITHM 2 becomes:

$$DES_{KEY_F}(v_{Fi}) = \frac{(\frac{v_{Wi}}{v_{Fi}} - 1)}{\alpha} \tag{4.12}$$

In formula (12), α is a fixed number, v_{Wi} is publicly known and is also fixed, therefore, the result of DES can be regularly derived from the right side of (12). This means that (i) if we do not know the key, we can break the DES because we can easily predict the result of DES (this contradicts the fact that the result of DES should be a random number sequence.); or (ii) if we can find the key KEY_F in reasonable time, then the known-plaintext attack is successful on DES (the cipher-text is $\frac{(\frac{v_{Wi}}{v_{Fi}} - 1)}{\alpha}$ and the plain-text is v_{Fi}). However, this contradicts the fact that, so far, there is no such an attack on DES known to the researchers. Therefore, as long as DES is safe, the above scheme is non-invertible.

Similarly, we can prove the non-invertibility of ALGORITHM 2 under DEFINITION 2.

Let us assume that ALGORITHM 2 is invertible with the consideration of DEFINITION 2. Then we can find v_{Fi} and w_{Fi} in reasonable time, such that

$$v'_{Wi} = v_{Fi} \times (1 + \alpha \times w_{Fi}), i = 1...n \tag{4.13}$$

where $(v'_{W1}v'_{W2}...v'_{Wn})$ is similar to $(v_{W1}v_{W2}...v_{Wn})$, and ALGORITHM 2 requires that

$$DES_{KEY_F}(v_{F1}v_{F2}...v_{Fn}) = (w_{F1}w_{F2}...w_{Fn}) \tag{4.14}$$

If we combine these two formulas together, then requirement of ALGORITHM 2 becomes:

$$DES_{KEY_F}(v_{F1}v_{F2}...v_{Fn}) = \frac{1}{\alpha}((\frac{v'_{W1}}{v_{F1}} - 1)(\frac{v'_{W2}}{v_{F2}} - 1)...(\frac{v'_{Wn}}{v_{Fn}} - 1)) \quad (4.15)$$

This means that, for any given sequence $(v_{W1}v_{W2}...v_{Wn})$, we can always produce a similar sequence $(v'_{W1}v'_{W2}...v'_{Wn})$. If we do not know the key, we can predict the results of DES as a stream regularly created by (15) (this also contradicts the fact that the result of DES should be a random number sequence.). If we can find the key KEY_F in reasonable time, then the known-plaintext attack is successful on DES (now the cipher-text is $\frac{1}{\alpha}((\frac{v'_{W1}}{v_{F1}} - 1)(\frac{v'_{W2}}{v_{F2}} - 1)...(\frac{v'_{Wn}}{v_{Fn}} - 1))$ and the plain-text is $v_{F1}v_{F2}...v_{Fn}$). Again, this contradicts the fact that there is no such an attack on DES known to the researchers so far. Therefore, as long as DES is safe, the above scheme is non-invertible. □

The Figure 4.5 shows the resulting frame applying our MPEG-based non-invertible watermarking scheme. By comparison, 4.4 shows the original frame.

Figure 4.4: A frame from the video clip "Hula" (the original).

4.4 Usage of Different Non-Invertible Schemes

DEFINITION 1 and 2 limit the non-invertibility to a single watermarking scheme, i.e., we assume that everyone uses the same watermarking scheme. However, there are many different

Figure 4.5: The watermarked frame by applying our MPEG-based non-invertible watermarking scheme, in which only AC coefficients were marked ($\alpha = 1$).

watermarking schemes available. What would happen if an attacker would use a different watermarking scheme than the true owner?

Suppose the owner uses a watermarking scheme (E, D, C), and the attacker uses a watermarking scheme (F, D', C') where $F \neq E$. Clearly, if the attacker uses an invertible scheme, then he/she can easily achieve $F(V_F, W_F) = V_W$. Therefore, we have to require that "a legitimate watermarking scheme must be non-invertible".

If (F, D', C') is non-invertible, then the attacker cannot claim the ownership of V_W by using F because the attacker can not find V_F and W_F such that $F(V_F, W_F) = V_W$. However, we notice that $F(V, W) \neq V_W$ also holds. This means that if the true owner applies F to his/her true original V and his true watermark W, he/she can not get V_W either. The verification process will deny the ownership to the true owner based on the fact $F(V, W) \neq V_W$ and based on DEFINITION 1 or 2. This statement triggers several questions: Should we disprove the ownership of the true owner based on $F(V, W) \neq V_W$? Do we need to modify the verification procedure discussed in previous section in order to allow using different watermarking schemes?

The answer is NO. The reason is that there is only one computationally feasible way to get V_W, that is by $E(V, W) = V_W$ and by knowing the key KEY. Therefore, it is sufficient that the verification procedure only verifies $E(V, W) = V_W$ in order to grant the ownership. We conclude our discussion with the following theorem:

THEOREM 4 Given any two different non-invertible watermarking schemes E and F, let us assume that the true owner A uses E and the attacker B uses F. Then we can grant the ownership to A unambiguously if and only if we can verify that $E(V,W) = V_W$.

PROOF:

The necessity is obviously true, i.e., if A is the true owner, then $E(V,W) = V_W$. Let us prove the sufficiency.

There are 4 cases which need to be verified:

$$E(V,W) = V_W? \tag{4.16}$$

$$E(V_F,W_F) = V_W? \tag{4.17}$$

$$F(V,W) = V_W? \tag{4.18}$$

$$F(V_F,W_F) = V_W? \tag{4.19}$$

Equation (17) is ALWAYS false because E is non-invertible. Equation (18) is ALWAYS false because $E \neq F$. Equation (19) is ALWAYS false because F is non-invertible.

Therefore, we only need to confirm whether Equation (16) is true or not. If Equation (16) is true, then A is the true owner. □

THEOREM 4 ensures that a real owner can be recognized and the ownership claim can be unambiguously resolved.

4.5 Conclusion

In this chapter, we presented a watermarking scheme for MPEG Encoded Video and proved its non-invertibility. The basic idea is using an encryption functions in the construction of the watermark. The same concept can be applied to many other schemes to achieve the non-invertibility. We provided a modified scheme as an example to show how and why the concept works and we discussed in depth the watermark construction requirements. We also discussed other issues such as the property of the Self-Proof Class, the choices of the "original", and the usage of different watermarking schemes, etc. Our results in Figure 4.5 show that, by applying our MPEG-based non-invertible watermarking scheme, we also achieved the goal of creating perceptually invisible watermarks.

The special construction of the watermark which we presented in this paper is the core to achieve the non-invertibility of the watermark scheme. In order to enforce these requirements, standardization and legislative procedure are also important. The standardization organization and the legislative bodies should limit the choices of the encryption functions within the well-known standard encryption functions, such as DES, IDEA, RC4 [51], etc. These encryption algorithms are widely used and can be considered safe. Otherwise, if someone claims that he/she uses an encryption function, how do we know that his/her encryption algorithm is really secure? The standardization organization and the legislative bodies should also limit the choice of eligible watermarking procedures within those widely recognized non-invertible schemes. Otherwise, if someone claims that his/her watermarking procedure is non-invertible, can we trust him/her? In summary, the research among watermarking schemes of video/image requires new technical solutions as well as the standardization and legislative procedures.

Chapter 5

Audio Watermarking

5.1 Introduction

While the basic concepts are easily applied to image or video in compressed domain and there are several algorithms designed for MPEG video streams, how to watermark MPEG audio streams is an interesting problem. There are three reasons which make us particularly interested in MPEG audio watermark: 1) Nowadays, the technology for audio applications is quite mature and audio streams are widely available throughout the Internet. In addition, audio data especially compressed audio data requires less resources (such as disk space, network bandwidth) than video data, it is much easier to copy, reproduce, and distribute. Therefore, to provide the copyright protection for audio data is urgent. 2) A lot of digital audio products are using MPEG audio format, therefore, the watermark should be added directly to the MPEG audio bit streams rather than applying watermark to uncompressed audio data and going through an expensive three-step (decoding-watermarking-encoding) process. 3) Because sound distortion is more sensitive to human ear than visual distortion to human eye, the watermark methods for MPEG video may not work for MPEG audio.

This chapter presents watermarking schemes on MPEG audio streams. We present two algorithms to embed watermark into MPEG audio streams. One algorithm deals with the scale factors of the MPEG audio data, and the other one embeds the watermark directly into the sample data. In order to be successful against the CMYY attack, as pointed out in Chapter 4 ([49]), the original MPEG audio streams are used in all of the watermarking processes -

watermark generation, watermark embedding process, and watermark verification. We also present the experimental results of our MPEG audio watermarking schemes.

5.2 Background – MPEG Audio

In this section, we will discuss the MPEG audio compression algorithm [48], [47] and the MPEG audio format. This serves as the necessary background to understand our MPEG audio watermarking schemes.

5.2.1 Overview

MPEG audio algorithm is an international standard for digital audio compression and does not make any assumptions about the nature of the audio source. It is suitable for audio-only applications as well as combined with video data (MPEG Systems Coding). MPEG audio compression is a lossy algorithm and uses the special nature of the human auditory system (HAS). It can provide a compression factor of 6-to-1 or more because it removes the perceptually irrelevant parts of the audio and makes the audio signal distortions inaudible to the human ear. MPEG audio standard defines the encoding process, decoding process, storage format, and three different layers of the coding system.

MPEG audio encoding process has the following steps: (1) Input audio samples are fed into the encoder. (2) The audio samples then pass through a mapping filter bank which creates subsampled representation of the input. The subsampled data is called subband sample because the filter bank actually divides the audio data into subbands of frequency. (3) At the same time, the input audio samples pass through MPEG psychoacoustic model which creates a set of data to control the quantizer and coding for each subband. The ratio of the signal energy to the masking threshold is used by the quantizer and coding block to determine how to allocate bit/noise information to minimize the audibility of the quantization noise. (4) Finally, the frame-packing block formats the quantized subband samples into a coded stream. Figure 5.1 shows the basic structure of an MPEG audio encoder.

MPEG audio decoding process is simply a reverse of the encoding process. The decoding process takes encoded bit stream as an input, puts it through the frame-unpacking block,

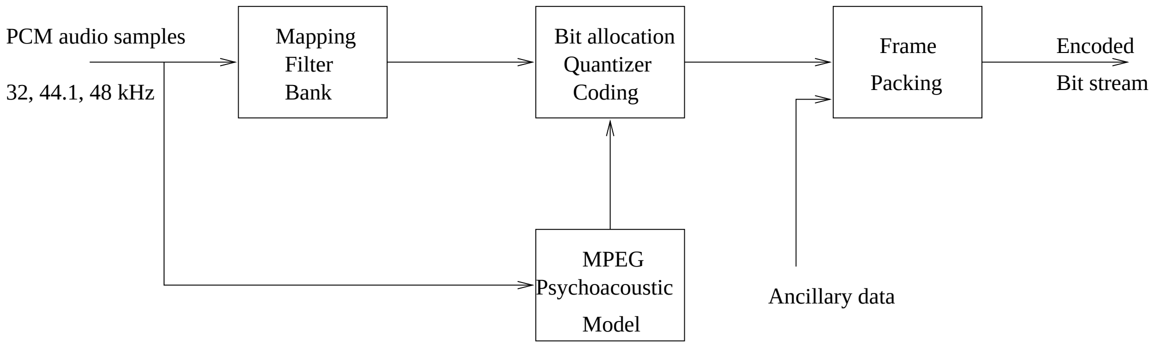

Figure 5.1: Structure of MPEG Audio Encoder.

reconstructs the frequency samples (subband samples), and then inverses the mapping to re-create the audio signal samples. Figure 5.2 shows the decoding process.

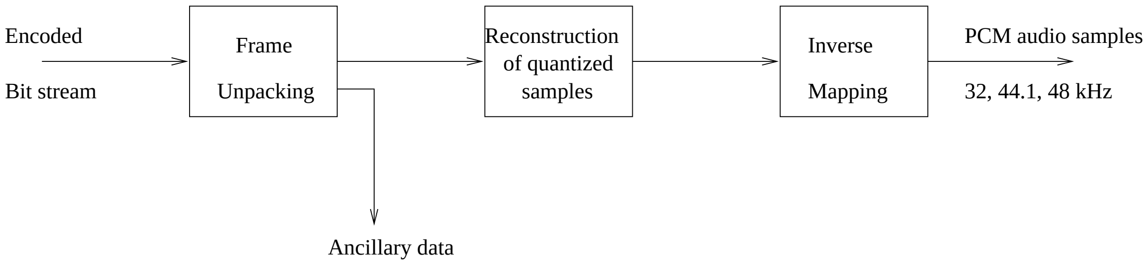

Figure 5.2: Structure of MPEG Audio Decoder.

There are different audio characteristics defined in MPEG audio standard, depending on different application requirements. MPEG audio supports three input audio sampling rates at 32kHz, 44.1kHz, and 48kHz. There are four different channel modes: 1) Single Channel - a monophonic mode; 2) Dual Channel - two independent audio signals coded within one bit stream; 3) Stereo Mode - left and right signals of a stereo pair coded within one bit stream; 4) Joint Stereo Mode - in addition to the Stereo Mode, this mode takes into account the stereo irrelevancy and redundancy between channels.

Depending on the applications, MPEG audio coding system can also be divided into 3 layers with increasing encoder complexity:

- Layer I contains the basic mapping of the audio samples into 32 subbands, fixed segmentation to format the data into blocks, a psychoacoustic model for the bit allocation, and quantization. It best suits the bit rate above 128 Kbps per channel.
- Layer II provides additional coding of bit allocation, scale-factors and samples. It targets the bit rate around 128 Kbps per channel.
- Layer III introduces increased frequency resolution based on a hybrid filterbank. It uses nonuniform quantizer and entropy coding (Huffman Coding). It offers the best audio quality at the bit rate around 64 Kbps per channel.

MPEG audio compression algorithm can have compression factors ranging from 2.7 to 24. The compression factors depend on different predefined fixed bit rates ranging from 32 Kbps to 224 Kbps.

5.2.2 Filter Bank

The filter bank divides the input audio signals into 32 equal-width frequency subbands and is used in all 3 layers described above. Although the filter bank is a lossy process, the information loss should be minimized and the audio distortion should be inaudible.

Notice that the HAS is best modeled by a set of 26 bandpass filters with bandwidth increases as frequency increases (these bands are called the *critical bands*). Hence the equal width subband model of MPEG audio does not accurately describe the HAS behavior. Especially, at lower frequencies, a single subband covers several critical bands. Because different critical bands have their own quantization bit numbers, we need one such number for the entire subband. In MPEG standard, the number of quantization bits for the entire band is decided by the critical band with the least noise masking.

5.2.3 MPEG Psychoacoustic Model

MPEG audio compression removes the perceptually irrelevant information of the audio signals by taking the advantage of the HAS' auditory masking ability. Masking is the effect that an audible signal becomes inaudible under a stronger audible signal. The masking effect depends on both temporal and spectral characteristics of the weaker and the stronger signals. The MPEG psychoacoustic model analyzes the audio signals and computes the amount of noise masking as a function of frequency. The encoder then uses this information to decide how to use limited number of bits to represent the input audio signals.

The MPEG audio standard has two psychoacoustic models which can be applied to any of the three layers. However, model 2 is more complex than model 1 and includes specific modifications for Layer III.

The MPEG psychoacoustic models have the following steps: 1) Calculate the spectrum - both models use Fourier transformation for the input sample to frequency domain conversion; 2) Identify tonal and non-tonal (noise) components; 3) Remove components which are below the absolute hearing threshold; 4) Find the masking threshold for each subband; 5) Calculate the signal-to-mask ratio.

5.2.4 MPEG Audio Frame Format

The MPEG audio stream consists of audio frames. Frame is the smallest unit which can be decoded individually. Each frame contains audio data, a 4-byte header, and possibly CRC (error detection code) and ancillary data. Frame header holds the basic structural information about the bit stream such as syncword[1], layer, bit rate, sampling frequency, mode, and so on.

The payload structures of Layer I and II are similar while the outlook of Layer III is different. Here we only provide details of the Layer II structure. Detailed information of Layer I and Layer III can be found in [48] or [47]. In Layer II, each frame has 1152 samples. Each subband has three groups of samples with 12 samples per group. The encoder can use a different scale factor for different groups of 12 samples as well as share scale factors among two or all three groups if the use of fewer scales does not increase distortion. The payload part of an MPEG audio frame has three parts: *SCFSI - SCale Factor Selection Information* field which holds the scale-factor selection information, *Scale Factor* field which holds scale factors for different samples, and *Sample* field which holds encoded sample data corresponding to 1152 samples of audio input. Figure 5.3 shows the format of the MPEG audio layer II.

[1] Loosely speaking, syncword can be thought of as the starting point of a new frame.

Header	CRC	Bit Allocation	Scale Factors	Encoded Samples	Ancillary Data

Figure 5.3: Frame Format of MPEG Audio Layer II.

5.3 MPEG Audio Watermarking Schemes

Audio data as well as other types of multimedia data are often stored and transmitted in compressed format, such as MPEG audio. The watermarking schemes should target the compressed data domain. Notice that, although the watermarking schemes such as Boney's scheme [14] work well in uncompressed data domain, extra decoding and re-encoding steps have to be taken if the audio data is already available in compressed format. Because the Boney's scheme uses the MPEG psychoacoustic model to mask the watermark, there would be no watermark presented in MPEG audio stream. Therefore, the study of watermarking in MPEG audio bit stream is interesting.

In this section, we cover the watermarking procedures for MPEG Layer II audio streams. However, the presented methods are easily extended to other layers of MPEG audio.

5.3.1 Watermark Creation

First of all, in order to be immune against CMYY attack, the watermarking scheme has to be non-invertible. There are several different approaches to implement the non-invertibility. One of them is to create the watermark by applying standard encryption algorithm such as DES to the original data. Here, we use the same method for the audio watermark creation. In [49] we proved the non-invertibility property for watermarking digital video and image. Similarly, we can prove the non-invertibility property for our audio watermarking schemes.

The watermark construction for MPEG audio has the following steps:

First, a key KEY is selected, and for each MPEG audio frame a_j, j=1, ..., N (number of audio frames), we apply DES with KEY to it to get a random byte sequence RBS:

$$RBS = DES_{KEY}(\text{one audio frame } a_j) \tag{5.1}$$

Second, let RBS_i be the i-th byte of the random byte sequence and w_i be the i-th bit of watermark bit stream, then the watermark can be created by:

$$w_i = \begin{cases} -1 & \text{if } RBS_i = \text{even number}, \\ 1 & \text{otherwise.} \end{cases} \tag{5.2}$$

The watermark bit sequence is applied repeatedly to the audio data of the same audio frame if the length of the watermark bit sequence (the number i) is less than the number of samples in a frame. Normally, i is in a range from 100 to more than 1000, therefore, it is secure enough to use it repeatedly. In the case that i is larger than the number of samples in a frame, the over-produced bits are ignored.

5.3.2 Watermark Embedding Procedures

Notice that there are two major parts in data fields of an MPEG audio frame. One is the *scale factors* and the other is the *encoded samples.* Both of these two parts can be used to embed the watermark. We describe the two procedures below.

5.3.2.1 Watermarking Scale Factors

Scale factor is the multiplier that makes the samples to use fully the quantizer range. The decoder multiplies the scale factor with decoded quantizer output to reconstruct the quantized subband samples. Each scale factor takes 6 bits, therefore, we have as many as 63 levels of scale factors (indexed from 0 to 62, 63 is not used by the standard). The changing scale factor has an auditory effect that the sound becomes stronger when the scale-factor level increases (index decreases) and becomes weaker when the scale-factor level decreases (index increases). However, tests show that a small change of scale-factor level (for example, increases or decreases by 1) normally can not be detected by the people. Our first MPEG audio watermarking procedure is based on this observation.

The watermarking procedure is very simple and just adds the watermark bit w_i to the index of the corresponding scale factor with two exceptions: (1) if index is 0 and $w_i = -1$, then do nothing; (2) if index is 62 and $w_i = 1$ than do nothing.

Let $ScaleFactor_i(index)$ be the i-th scale factor with the level indicated by $index$ and $ScaleFactorW_i$ be the i-th watermarked one. The watermarking procedure can be described as:

$$ScaleFactorW_i = \begin{cases} ScaleFactor_i(index) & \text{if } index + w_i = \text{-1 or 63} \\ ScaleFactor_i(index + w_i) & \text{otherwise.} \end{cases} \tag{5.3}$$

This scheme has several drawbacks. The first one is that, for some audio streams, there are only a few scale factors for a frame, therefore, our frame-based watermarking scheme does not have much data to watermark. One solution is to group frames together and to apply the watermarking procedure at a group of frames.

The second drawback is the level value added/subtracted to/from the scale factor as a watermark. When the scale factor is increased by 1 level (index decreases by 1) or decreased by 1 or multiple levels, there is no significant audio distortion. However, when the scale factor is increased by 2 levels or more, there is often perceivable audio distortion and the noise can be heard. This introduces a problem that multiple watermarks can not be applied. The reason is that when multiple watermarks are applied, certain scale factors would be increased by multiple levels and perceivable noise would be introduced.

The third drawback follows from the second problem: an attacker can lower the scale factors dynamically by 2 levels or 3 levels (increase the index by 2 or 3) and destroy the watermarks.

5.3.2.2 Watermarking Encoded Samples

The other choice is to embed the watermark into the sample data. The basic idea is to add the watermark (-1/1 bit sequence) to encoded sample sequence. However, changing these encoded samples is very sensitive. Tests show that, if we change every encoded sample by either adding 1 or -1, the distortion of the resulting audio is easily detected by human ear and the watermarked audio is not acceptable.

In order to solve this problem, we introduce a *spacing parameter sp* which has the following meaning: in approximately every *sp* samples, we randomly select 1 or 2 samples to watermark. By choosing a good spacing parameter, the distortion can be minimized. This conclusion is also supported by our experimental tests (see next section). The use of spacing parameter gives us a method to adaptively watermark MPEG audio depending on different audio streams.

The watermark creation procedure is slightly modified to incorporate the spacing parameter:

$$w_i = \begin{cases} -1 & \text{if } RBS_i = 0 \text{ (mod sp)}, \\ 1 & \text{if } RBS_i = 1 \text{ (mod sp)}, \\ 0 & \text{otherwise.} \end{cases} \tag{5.4}$$

The watermarking procedure is similar to the previous one except that we have to make sure that the watermarked sample does not have the format of "111...1" because this kind of sample coding is illegal in MPEG audio standard.

Let $Sample_i$ be the i-th sample in an audio frame and $SampleW_i$ be the i-th watermarked sample. Let $nbal_i$ be the number of bit allocation for i-th sample (the information of bit allocation for each sample comes from the Bit Allocation field in an audio frame). The watermarking procedure can be described as:

$$SampleW_i = \begin{cases} Sample_i & \text{if every bit of } (Sample_i + w_i) \text{ is } 1, \\ Sample_i + w_i & \text{otherwise.} \end{cases} \tag{5.5}$$

5.3.2.3 Discussion

Both MPEG audio watermark schemes are simple extensions of the concepts from spread spectrum communications. They are designed to apply watermark in the compressed data domain so that the expensive decoding/re-encoding can be avoided.

Because the creation of the watermark is based on applying standard encryption function to the original audio data, the non-invertibility of these schemes can be easily proved. We omit the proof here because it is similar to the proof of our non-invertible watermark scheme for video and image streams as shown in Chapter 4 [49].

Another requirement is that the original MPEG audio stream must be presented in the verification process because, in general, the watermark schemes which do not use the originals in their verification are invertible.

Finally, we want to point out that, because the human ear is more sensitive to audio distortion than human eye to image distortion, the amount of data which can be embedded as watermark is very limited and depends on the content of the audio streams. For this reason, we introduced in the second scheme the spacing parameter which can be thought of as an indicator to measure how much data can be embedded. By choosing a smaller spacing parameter, we can reach the upper limit of auditory acceptance. This means that if we embed more data, the distortion will be noticeable. This implies that multiple watermarks are difficult to apply. The spacing parameter is also required in the verification process.

5.4 Experimental Tests and Results

In this chapter, we only focus on the experimental results regarding the sound distortion caused by adding a watermark. Other types of experimental tests, such as watermark detection, are common among all watermark schemes and have been discussed extensively in other publications.

Our tests use 5 different audio clips, and they are all MPEG Layer II audio data. Table 5.1 shows the detailed information about all clips. The content of the audio clip "Harmonic_Ocean.mp2" is a segment of contemporary music (light music), "t3ac.mp2" is an audio clip of classical music - a segment of Symphony, "lsclip1.mp2" audio clip includes a soprano aria, and "isochrShaping.mp2" and "klara1.mp2" audio clips are lecture talks. We try to use short clips because it is always difficult for testers to memorize longer clips and then compare their original and watermarked versions.

Our tests involve 7 people with different music and technical background, 4 of them have music background such as playing piano or singing. Both presented watermarking schemes are

Audio Clip Name	Type	Sampling Frequency	Frame Size (Byte)	Total Frames	Playing Time	Bit Rate
Harmonic_Ocean	Layer II Stereo	44.1kHz	626-627	1729	45.16s	192 kb/s
t3ac	Layer II Stereo	44.1kHz	1253-1254	803	20.97s	384 kb/s
lsclip1	Layer II Stereo	32kHz	288	552	19.87s	64 kb/s
isochrShaping	Layer II Single Ch.	44.1kHz	104-105	3960	103.44s	32 kb/s
klara1	Layer II Single Ch.	44.1kHz	104-105	5136	134.16s	32 kb/s

Table 5.1: Basic Characteristics of Tested Audio Clips.

Audio Clip Name	Scale Factor ± 4	Scale Factor ± 3	Scale Factor ± 2	Scale Factor ± 1
Harmonic_Ocean	7	4	1	0
t3ac	7	5	1	0
lsclip1	7	5	2	0
isochrShaping	7	6	1	0
klara1	7	6	3	1

Table 5.2: Testing Results on Watermarking Scale Factors. Each entry in the table indicates, out of 7 testers, the number of testers who notice difference between the original and the watermarked audio clips.

tested on each audio clip individually. In each test, the tester is given the original, watermarked, and over-watermarked audio clip. The over-watermarked data are created by increasing/decreasing 2 or more levels on scale factors in *Scale Factor Watermarking* scheme and by choosing smaller spacing parameter in *Audio Sample Watermarking* scheme.

Table 5.2 shows the testing results on watermarking Scale Factors. Each entry in the table indicates, the number of testers out of 7 testers who notice the difference between the original and the watermarked audio clips. When there are small changes (± 1) applied to the scale factor level, no one notices the difference (with one exception). When the level changes are larger, some listeners start to notice the distortion. For example, when testing the audio clip "lsclip1.mp2", 5 out of 7 testers notice the distortion when scale factor changes to level ± 3.

Audio Clip Name	SP 2	SP 4	SP 6	SP 8	SP 12	SP 16	SP 20	SP 30	SP 40	SP 60	SP 80
Harmonic_Ocean	2	0	0	-	-	-	-	-	-	-	-
t3ac	4	1	0	0	-	-	-	-	-	-	-
lsclip1	-	-	-	7	6	6	6	5	4	1	0
isochrShaping	-	-	7	6	5	4	2	2	1	0	-
klara1	-	-	7	6	5	5	2	2	1	0	-

Table 5.3: Testing Results on Watermarking Samples. Each entry in the table indicates, out of 7 testers, the number of testers who notice difference between the original and the watermarked audio clips. The "-" entry means no testing is conducted.

Table 5.3 shows the testing results on watermarking MPEG audio samples. Each entry in the table indicates the number of testers who notice the difference between the original and the watermarked audio clips. The "-" entry in the table means no testing is conducted. We can see that the more samples are watermarked (i.e., the smaller SP parameter is), the easier it is for the testers to notice the sound distortion. We also notice that the amount of data, which can be embedded without significant distortion, depends on the content and the nature of the tested audio clips. For example, for audio clip "Harmonic_Ocean", we can easily mark 2 samples out of every 4 audio samples without knowing distortion. On the other hand, it is very sensitive to watermark audio clip "lsclip1".

5.5 Conclusion

In this chapter, we presented watermarking methods which embed the watermark directly into the MPEG audio bit streams. This is very much desired if the audio streams are already in MPEG encoded format. Otherwise, if we use watermarking schemes designed for uncompressed audio, we have to go through the expensive decoding-watermarking-encoding process. Between two presented schemes, one adds the watermark into the Scale Factors in MPEG audio streams and another embeds the watermark into the MPEG encoded samples. In the design of our second scheme, we have introduced a *spacing parameter* which actually provides a way to adaptively watermark MPEG audio streams.

Our experimental tests show that both methods minimize the distortion and provide good audio quality for various types of audio streams. Although we use only MPEG Audio Layer II streams in our experimental tests, both of the proposed schemes can be applied to MPEG Audio Layer I streams and Layer III streams. We also take the rightful ownership problem into our consideration. By creating the watermark using a standard encryption function such as DES, the non-invertibility property can be achieved and the unique ownership can be determined.

Chapter 6

Customer's Rights

6.1 Introduction

Another open problem in using watermarks refers to the customer's right protection. The *rights of the legitimate customers/buyers* when using watermark have not been addressed at all. Currently, owners/sellers of video/image information have complete control of the watermarking procedure, and customers have to accept the transaction from the owners (sellers) without any certificate to prove their rights to use the watermarked data. Without solving this problem, the use of watermarked data is limited especially in the Internet environment. More seriously, without addressing this problem properly, the goal of using watermarks to protect the owner's copyright could eventually be defeated.

In this chapter, we present two protocols to provide protection of customers' rights. We introduce a certificate/ticket information for the customer. The customer's certificate needs to be tied into the watermarking process at the owner's side. Although the watermarking process is still performed by the owner, only the customer knows the key which creates the certificate. Once the customer purchases the watermarked data, he/she can prove his/her rights by showing that the certificate is indeed created by his/her private key. The secure exchange of the certificate and watermarked data is performed by our protocols.

6.2 Customer's Right Problem

Until now, watermarking schemes involved only the owner of the video/image information. The owner had complete control of the watermark construction, insertion of watermark into video/image, and its verification. All existing watermarking schemes are designed to protect the original owner's copyright rather than the customer's right. Currently, to the authors' knowledge, there is no effort to even identify the customer's right problem when using watermarked multimedia data.

There is an implicit assumption, shared by all schemes, that the owners (or the holders of the originals) themselves are always trustworthy. However, this is not always true. Because of the above implicit assumption, all watermarking schemes are bias and unfair to legitimate customers. Legitimate customers are those who have the right to access the watermarked information through legal ways such as purchasing, signing agreement, etc.

We will give examples to show that these problems exist. Notice that, the bias not only is unfair to the customers but also causes chaos and eventually defeats the purpose of using watermarks to protect the owner's copyright. The reason will be clarified in the following examples.

EXAMPLE 2 Suppose owner O mistakenly sells the same version of a watermarked image V_W to customer B and C. Legally only B and C should have V_W. Let us assume that B then illegally gives V_W to another person D. Because B and C have the same version, who is responsible for giving V_W to D? Even if O gives B and C different watermarked image, B could argue that O made a mistake and gave C the same version (therefore, it is possible that C has two versions, one of them is the same as B's), hence B is not responsible. B can also argue that O gave the same V_W to D. How does C and/or O prove their innocence? How can C and/or O prove that B is responsible for the illegal transaction? □

One may argue that such mistakes can be easily avoided and no owner would intentionally make such mistakes. Let us look at another example.

EXAMPLE 3 Owner O hires Agent A to distribute image V for O. Legally, A should give different customers B1, B2, ..., B99, and C different watermarked image $V_{W_{B_1}}$, $V_{W_{B_2}} \ldots$, $V_{W_{B_{99}}}$,

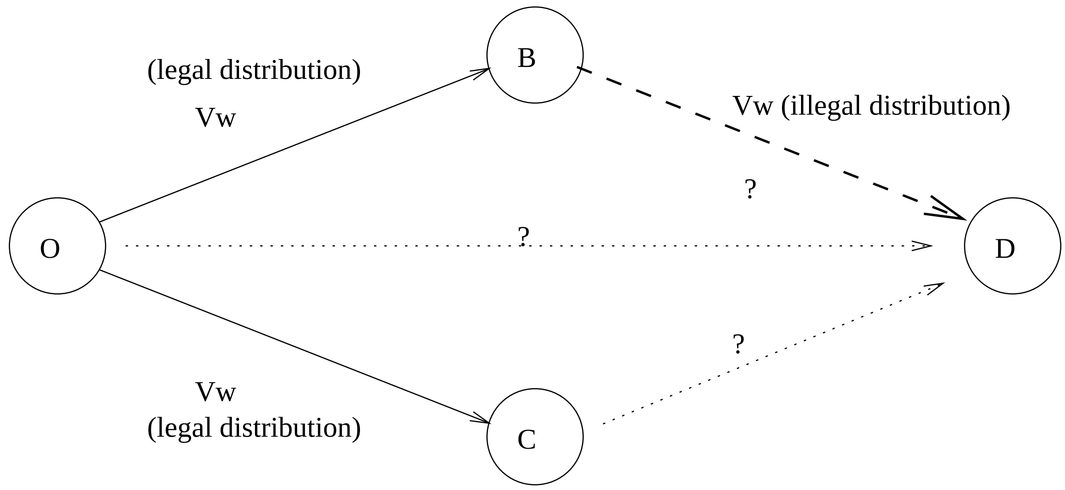

Figure 6.1: EXAMPLE 2.

and V_{W_C}, respectively, and give O royalties on the per-copy bases. However, A could illegally sell the same version watermarked using C's watermark W_C to C and to B1, B2, ... B99 and then claim that he/she only sold one copy to C. C would be blamed for other 99 copies because these copies have only C' watermark on (We can assume that some or all of B_i's may not be trusted.). Clearly, C is a victim in this case. How can C protect his right, and prove his innocence? □

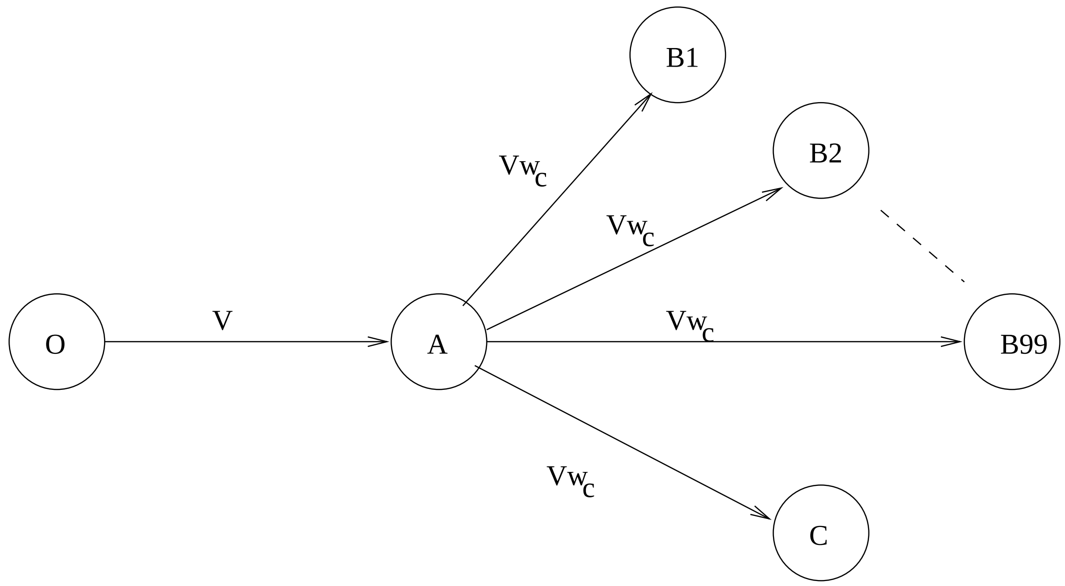

Figure 6.2: EXAMPLE 3.

EXAMPLE 4 Currently, the use of watermarked multimedia data is limited, especially in the Internet environment. Even if C purchases the right of V from O and gets V_{W_c} which is unique and specifically designed for C, C can not use V_{W_c} publicly such as putting the image on C's web page. If C does so, he would lose the control of V_{W_c} and his right on V_{W_c} because other people can download V_{W_c} from C for free. If B downloads V_{W_c} and starts to sell it illegally, C will be wrongfully accused instead of B because V_{W_c} has only C's identity on it and C has no way to prove his innocence. Would C be able to use V_{W_c} publicly and at the same time prevent others' illegal use? □

In the above examples, it is difficult to decide during the verification process whether, the owner, agent, or customers acted illegally. Current watermarking schemes, locally and completely controlled by the owner or the creator of the watermark, cannot guarantee copyright protection of customers/buyers, therefore do not provide a solution to the customer's right problem. Furthermore, the owner's copyright would not be protected because the owner would not be able to identify the illegal party (although the ownership can be decided).

In summary, we should not by default trust the owner or creator of the watermarked data. A copyright protection scheme, using and exchanging watermarked data, should be fair to all legal transaction participants and protect both true owner's and legitimate customer's rights. Therefore, we need new protocols which involve both the owner and the customers to accompany the existing watermarking services.

6.3 Customer's Right Protection Protocols

The basic idea behind our protocols is to provide a unique certificate/ticket information for each customer and embed it in the watermark. The obvious requirement is that the owner should not be able to get the keys which are used to create these certificate/ticket information. Otherwise, it is equivalent to the existing (local) watermarking schemes. There are two possible approaches to do this. One approach is to use a trusted third party (TTP), in which the TTP will handle (1) watermarking processes for all owners and customers and (2) the watermark verification process. The other approach is to design a protocol to handle watermarking processes between the owner and the customers. TTP will be used only in watermark verification process.

6.3.1 Notation

We will adopt, as much as possible, the common notation in the literature. In this chapter, O and A are the owners (senders, sellers) where A could also serve as an agent. B, B_i, and C are customers (receivers, buyers). TTP is a trusted third party, we also use S to represent the trusted third party.

$KeyA, KeyC$ are private keys known only by A, C, respectively. $KeyOC$ is a private key known only by O and is generated specifically for C. KSA, KSC, and KSO are private keys between TTP (S) and A, C, and O, respectively. KA is a key and KA^{-1} is its inverse. In symmetric cryptosystems $KA = KA^{-1}$. In asymmetric cryptosystems, we tend to use KA as the public key and KA^{-1} as the secret key of the pair (KA, KA^{-1}). $\{M\}_K$ means that message M is encrypted with key K. If K is a secret key, $\{M\}_K$ also means that M is signed by K. Notice that, we use terminology "private key" in symmetric cryptosystems and "secret key" in asymmetric cryptosystems.

"$P : J$" means that a participant P performs a computation job J. "$A \longrightarrow C : M$" means that A sends a message M to C. V is the original multimedia data, V_W is the watermarked version. C_0 and C_1 are seeds to create watermark. The size of C_0 and C_1 are the same as the original V, i.e., each bit of C_0 and C_1 corresponds to one pixel of V. $\oplus$ represents the watermarking process.

6.3.2 TTP Watermarking Protocol

TTP Watermarking Protocol uses a Trusted Third Party in the middle between the owner O and the customer C. O and C do not talk to each other directly. If O wants to send a watermarked data V_W to C, O sends the original V to TTP. TTP then creates V_W for C using a non-invertible watermarking scheme, makes sure that the watermark used is unique for C, and sends V_W to C. Here, we assume that all information from O to TTP and from TTP to C are transmitted through a secure channel. We also assume that all necessary setup steps such as authentications, key distributions, and link setup are done properly in advance. TTP could perform the watermarking process (including the verification) solely or could be combined together with the security service server such as the key server.

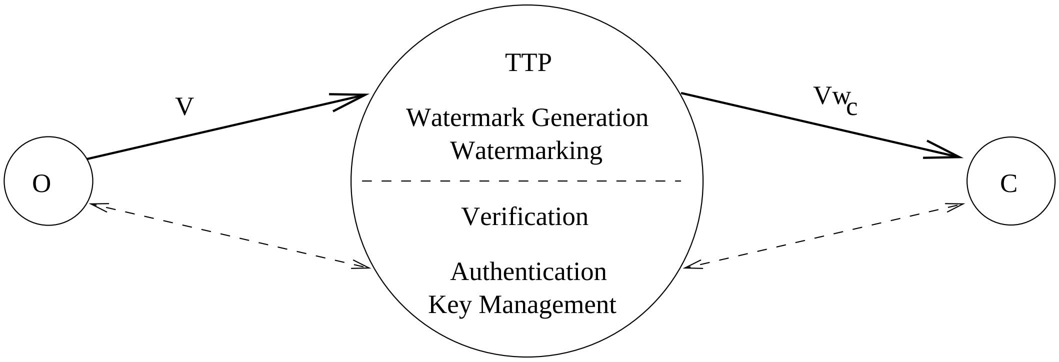

Figure 6.3: TTP Watermarking Protocol

The TTP Watermarking Protocol has the following steps:

Step 1. O —> S: $\{V\}_{KSO}$

Step 2. S: $V_W = DES_{KeyOC}(V) \oplus V$

Step 3. S —> C: $\{V_W\}_{KSC}$

The advantages of the TTP Watermarking Protocol are: (1) the problems we discussed in Section 4.3 can be avoided. TTP can make sure that each distributed copy has unique watermark in it. To verify whether a user a legal customer or not, TTP simply searches the records kept by TTP. If there is a record that matches user's name (user's name must be authenticated), then he/she is a legal user, otherwise, he/she is an illegal user. (2) it can provide standardized schemes for users. This is very important in order to achieve the non-invertibility in watermarking services.

The disadvantages of TTP Watermarking Protocol are: (1) TTP does too much work hence it can become a performance bottleneck. (2) TTP has all originals. This means that if TTP is compromised, not only the customer's right can not be protected but also all originals are exposed. (3) TTP has to do book-keeping for all originals and evidence for all users. Therefore the size of TTP's data base will be too large. (4) Other common problems shared by all centralized schemes can occur. For example, if TTP is not available, then no one can deliver the watermarked data.

6.3.3 Owner-Customer Watermarking Protocol

Owner-Customer Watermarking Protocol allows individuals to use watermarking services and exchange data among themselves. TTP is still needed but its role is reduced to verification purposes. In order to get a unique watermarked version, C needs to provide its own evidence of identification to A to create the watermark. A then uses C's information to create the watermark, and applies a non-invertible watermarking scheme. The problem is that A can see the information provided by C therefore A could modify, destroy, or replace C's identification information.

One way to solve this problem is to have C apply a standard encryption function such as DES to some seed information C_0 using secret key $KeyC$, and give the encrypted message to A as the base for creating the watermark.

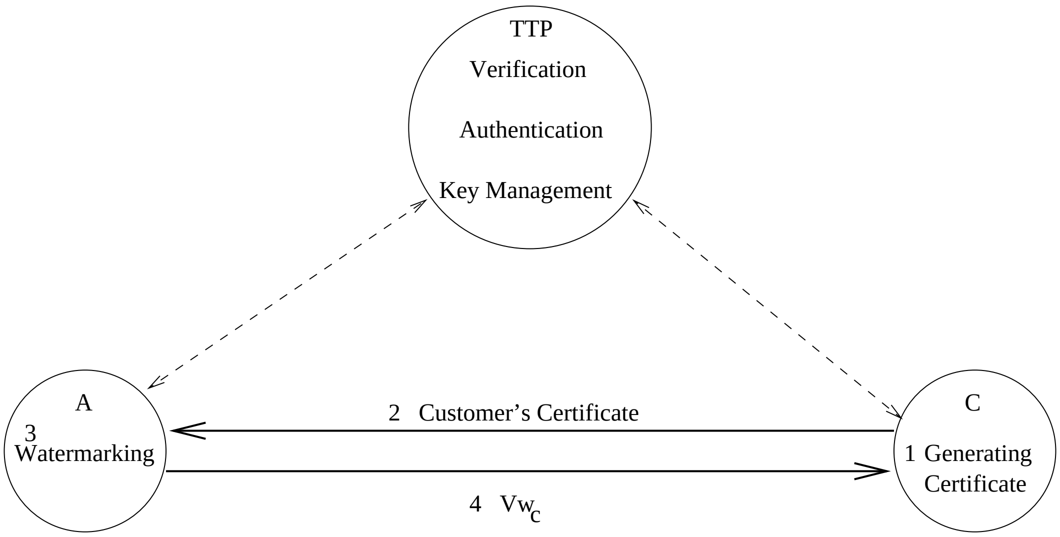

Figure 6.4: Owner-Customer Watermarking Protocol

The Owner-Customer Watermarking Protocol has the following steps:

Step 1. C: $C_1 = DES_{KeyC}(C_0)$

Step 2. C —> A: $\{\{C_1\}_{KC^{-1}}\}_{KA}$

Step 3. A: $V_W = DES_{KeyA}(C_1 \ Xor \ V) \oplus V$

Step 4. A —> C: $\{\{\{V_W\}_{C_1}\}_{KA^{-1}}, \{KeyA, V, C_1\}_{KSA}\}_{KC}$

6.3.3.1 Protocol Discussion

Now let us discuss each step of the Owner-Customer Watermarking Protocol in detail.

(1) C uses DES to create its own identification certificate C_1 which is a random bit sequence. Without knowing $KeyC$, no one can create the same certificate C_1 (not even a certificate similar to C_1. This is guaranteed by the randomness of the results created by DES). The size of C_0 should be the same as the size of image V which will be watermarked, i.e., each bit of C_0 corresponds to one pixel of the image. This comes from the way we design our non-invertible schemes in Section 3.4. Notice that, although there are many ways to select C_0, C_0 should not be an arbitrary sequence. If C_0 could be anything, then B could choose any K_{b0}, compute $B_0 = DES^{-1}_{K_{b0}}(C_1)$, and destroy the uniqueness of C_1's creation because C_1 also satisfies $C_1 = DES_{K_{b0}}(B_0)$. The choice of C_0 could be some publicly known number sequence such as the first n digits of π. The choice of C_0 should be mutually agreed between A and C in setup phases. The actual size of V is also exchanged in setup phases. The key $KeyC$ will not be sent to anyone (including TTP) and is known only by C. Only when verification is needed, $KeyC$ is provided to TTP.

(2) C signs its evidence with the secret key KC^{-1} and then encrypts it using A's public key KA and sends it to A. How to make Step 2 secure is not the focus here. There are many schemes and protocols [45, 1, 10] which can successfully handle this concern. Some auxiliary information, such as the identification of the requested multimedia data, may also be needed. Notice that we assume that all necessary authentication and key distribution steps have been done in advance.

(3) A uses C_1 as the base to generate a watermark for C, applies a non-invertible watermark scheme to the original V, and creates the watermarked image V_W. $KeyA$ is also known only to A and will not be sent to anyone including TTP, except when the verification is needed. The use of non-invertible scheme is for protecting A's copyright and is not for C' right protection.

(4) A sends $\{\{V_W\}_{C_1}\}_{KA^{-1}}$ to C. By encrypting V_W using C_1, (i) A vows that A indeed created the watermark using C_1, and (ii) C vows that C knows C_1, and can therefore get V_W. (i) is important because otherwise A could use, in Step 3, C_1' which C does not know. Then C could not prove that C has right on V_W. This could be viewed as binding the information C_1 with V_W. The reason we can use C_1 to encrypt V_W is because C_1 is a random bit sequence created by DES and known only by A and C. Simple XOR operation of C_1 with V_W is sufficient for this purpose. (ii) is important because otherwise C could claim that C did not provide C_1 to A (This can also be avoided with a careful design in Step 2. For example, C binds C_1 with the identification of V but A needs to keep the binding information. Because (ii) can solve this problem, we do not need the binding in previous steps.). The second part of the message, i.e. $\{KeyA, V, C_1\}_{KSA}$, is optional and is used for the verification purpose only. KSA is the key known by A and TTP. $KeyA$ and the original V won't be exposed until the verification is needed. The same information (i.e., $\{KeyA, V, C_1\}_{KSA}$) can be provided by A when the verification is needed if we do not include this part in the protocol.

6.3.3.2 Verification Procedure

There are two issues involved in the verification procedure. One issue is that C needs to verify whether or not V_W is actually created by using the Owner-Customer Watermarking Protocol. C needs this verification to be sure that C has the proper right to own V_W. However, C does not have to verify this immediately. Another issue is to decide who is the legitimate customer when someone other than C has the same V_W. The verification procedure must be processed using the TTP.

For the verification purpose, C sends all evidence to TTP, i.e.,

C —> TTP: $\{\{\{V_W\}_{C_1}\}_{KA^{-1}}, \{KeyA, V, C_1\}_{KSA}, KeyC, C_0\}_{KSC}$

Notice that C_0 is known by both A and C.

- To verify if V_W is created according to the Owner-Customer Watermarking Protocol, TTP verifies the formula used in Step 3.

- To verify if C is the legal customer of V_W, TTP verifies that (1) C_0 is the seed information agreed upon between A and C and (2) $C_1 = DES_{KeyC}(C_0)$. Clearly, if B holds V_W for whatever reason, but does not know $KeyC$, then it is computationally impossible for B to choose $KeyB$ such that $C_1 = DES_{KeyB}(C_0)$. In this way, the uniqueness of C' right will be established.

6.4 Further Discussion on The Protocols

6.4.1 The Role of TTP

Both the TTP Watermarking Protocol and the Owner-Customer Watermarking Protocol need a trusted third party. TTP acts as the judge and performs the verification in both cases. The difference is that in the Owner-Customer Watermarking Protocol, TTP's role is reduced and the actual watermarking process is done at the owner/agent side. With the Owner-Customer Watermarking Protocol, TTP does not have to keep all information, once the verification process is finished, TTP can delete those data.

TTP can also acts as the authentication and key server. Otherwise, we need a separate server to handle these functions.

6.4.2 Auxiliary Steps and Messages

Just like any secure network protocols, the procedures of the key distribution and authentication are needed before the actual information exchanges between owner A and customer C. There exists an extensive related work as described in Section 4.2. We assume that all necessary authentication and key distribution protocols are performed properly in advance. The details of these protocols are beyond the scope of this paper.

Extra messages are needed. For example, the party's name should be included in the messages, and nonce identifiers which can provide the proof of a message's freshness are also useful. In order to emphasize the new aspects of our protocols, we have not put these details in our protocols.

There are other concerns when the protocols are actually implemented. For example, the non-repudiation protocols should be considered in order to validate irrefutable evidence regarding the transmitted messages between participants. This is particularly important if selling and buying transactions are involved.

6.4.3 Solutions to Our Examples

With the TTP Watermarking Protocol, the problems presented in EXAMPLE 2, 3, and 4 can be solved easily. TTP simply checks whether a user has a purchasing record in TTP's database or not. If there is a record, then the user is a legal customer, otherwise, he/she is not.

Let us check how the problems presented in EXAMPLE 2, 3, and 4 can be solved with the Owner-Customer Watermarking Protocol.

The problems in EXAMPLE 2 and 3 can be solved because, with the Owner-Customer Watermarking Protocol, all legal customers hold their own evidence to prove the customer's right. Neither the owner/agent nor other users know the evidence. Thus no one other than the legal customers can prove their own rights. This prevents the illegal distribution and the distribution of the same version by mistakes.

The problem in EXAMPLE 4 can also be solved. Using the Owner-Customer Watermarking Protocol, C has the freedom to publicly use V_{W_c}. B can still download V_{W_c} from C, but B can no longer sell it. Buyer of B could verify if B has the proper right by asking B to prove it to the judge (TTP). B can not even use the free downloaded V_{W_c} publicly, because B can not provide necessary evidence to prove his customer's right.

6.5 Conclusion

In this chapter we presented two solutions to the copyright problem which arises for the customer/buyer of a video/image/audio. The customer/buyer must protect its own rights and must be able to prove that he/she purchased it from the owner of the video/image in case of a verification request during a legal process.

Both protocols incorporate the basic idea that when a video/image information is watermarked for the customer/buyer, the customer's unique identification is encoded in the watermark and the key to create the identification is not known to the owner of the original video/image. This approach ensures that only the legal customer can prove that he/she purchased the video/image by showing his/her certificate. As a result, the customer has full freedom to use his/her legal copy. In both protocols we rely strongly on a trusted third party in case of a verification request during a legal process.

Chapter 7

Multimedia Security in Mulicasting Environment

7.1 Introduction

We present simple, efficient, and secure multicast protocols with copyright protection in an open and insecure network environment. There is a wide range of multimedia applications that can benefit from using our secure multicast protocol such as the commercial pay-per-view video multicast or highly secure military intelligence video conference.

The existing multicast security protocols [5, 44, 66, 43] are based on the general concept that the senders encrypt the data using a *group key* and then they multicast the encrypted data to the group members. Upon receiving the encrypted data, the receivers decrypt the data using the group key. The essence of the multicast security protocols lays in the *key distribution scheme* which distributes the group key to the group members in a secure, efficient, and scalable manner. However, there are many other important issues which are not addressed or only partially addressed by the existing key distribution protocols. We will describe these issues as follows.

- *Open Network Environment*

  We assume that the content providers (senders) and the customers (receivers) are connected through an open network environment. This means that the multicast stream may have to travel through intermediate switches or routers which may or may not have any security mechanism. Therefore, our secure multicast protocol must not depend on any of the intermediate network components for security support. For example, the Scalable Multicast Key Distribution by Ballardie [5] does not satisfy this property. It is based on the Core Based Tree architecture [4, 6] which requires the network routers to maintain a hard state, and it depends on the Core routers to maintain security information.

- *Robust Dynamic Group Membership Support*

  Packet loss or long network delay is prevalent in any open network environment such as the Internet, where the traffic congestion level and bandwidth availability for members of the same multicast group can vary significantly. As a result, the key distribution protocol must deal gracefully with lossy or long delay multicast channels. For example, the group key management protocol by Wallner[66] uses the same multicast channels to distribute the group key as well as the multicast data. It is very likely that some members in the multicast session do not get the *rekey* messages in time because of data loss or long network delay. This can create windows of security vulnerability. For example, a newly expelled receivers may still be able to decrypt the multicast data from senders who did not receive the *rekey* messages in time. Although retransmission(s) and acknowledgment message could be used to solve the data loss problem, the bandwidth overhead caused by retransmission(s) and the associated delay in acknowledgments are unacceptable, especially for multimedia applications that have real time constraints, such as video/audio distribution.

- *Efficient Support for Dynamic Group Membership*

  Given a large group size, the membership changes can occur frequently. This will result in a flooding of *rekey* messages in the multicast channel and the associated processing overhead to decrypt the sequence of keys at the member's side.

  We allow the customers to join or leave the multicast session at any time. After the customers leave the multicast session, they should not be able to decode the multicast stream. After new customers join the multicast session, they should be able to decode the multicast data (but not the data before they join). In addition, we also allow the multicast session leaders (e.g., the content providers) to remove or temporarily block out any customers in the session (for reasons such as parental control). The group key management protocol must support join/leave/expel operations in an efficiently manner with minimum delay. At the same time, these operations must not create any window of opportunity for security violations in the multicast session and they must not cause any service disruption to other customers.

- *Copyright Protection*

  We assume that the content providers need to have copyright protection for multicast video data, so that the rightful ownership of the multicast video can be identified. We apply the watermark technique to encode the ownership information into the video data.

  It is possible that some insiders (paying customers) in the multicast session, may distribute the multicasted data to the outsiders for free or for a small profit. The leaking of this multicast stream is a copyright violation, and the consequences can be severe. Potential customers may be less likely to join the multicast session for a full price, given that they can get the same but illegal copy from the leakers for free or at a cheaper price. Our multicast protocol enables the content providers to identify the leakers in the multicast session after the leaking copy is discovered and its watermark analyzed. This is a preventive method. Knowing that the content providers can uniquely identify the leakers in the multicast session will prevent unscrupulous people from actually carrying out the act of leaking. This feature is very useful in a multicast session involving military intelligence agents to help to identify a spy.

In the following sections, we will show that our protocols solve the problems mentioned above.

7.2 Stream-carried Key Distribution

Any secure multicast scheme must consider the key distribution problem. Currently, all existing schemes assume either the key distribution is done in session setup stage or at least the key is not changed too frequently. None of the existing schemes can handle the dynamically changed membership. Furthermore, multimedia data such as video streams has strict time constraints, therefore, any membership changing information and new keys must be delivered in a timely fashion. All existing schemes certainly do not satisfy this requirement.

In this section, we present our schemes to achieve the above requirements for the multimedia data.

7.2.1 Key Distribution Algorithm

Assume that a user can join or leave at any time (= at any video frame) during a multicast session, and a new key must be used for that frame. To avoid extra synchronization efforts between video frames and their corresponding keys, it is better to transmit the new key with the associated video frame. In order to distribute the keys together with the data stream, we need to define a special header to carry the keys. The method for key distribution has the following steps:

1. Create pairwise key between sender and each receivers. These keys can be either symmetric (private key system) or asymmetric (public key system) and can be done during the user authentication process. How to deal with the authentication process is not the issue here, there is an extensive research in that area and many reliable schemes are available.

Suppose we have n users, we denote $k_1, k_2, ..., k_n$ as the pairwise keys for User 1, 2, ..., n, respectively.

2. Sender chooses a data encryption key KEY which is used to encrypt the actual MPEG video data.

3. The encrypted MPEG stream with the special header to carry the keys have the following structure:

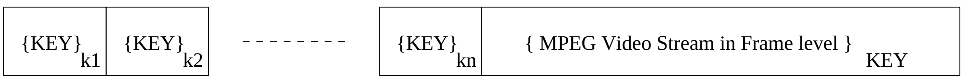

Figure 7.1: Key Header and Encrypted Stream Structure

4. Whenever a new data encryption key KEY' is needed, this special header is inserted into the MPEG stream, and the following MPEG stream is encrypted using new key KEY' until next time it is changed.

5. When user i receives the encrypted stream, she looks for the special header first, if the header is presented, she will looks for i-th slot, decrypt the message and get KEY, and then she will be able to decrypt the MPEG stream using KEY.

Although this is a simple, brute-force method to distribute the keys to end users, it has the following advantages:

- Security: this method achieves the highest possible security level. It does not depend on the intermediate nodes, or core routers for security, therefore, there is no need for doing intermediate node/core router authentication.
- Protocol independence: it can be applied to any multicast schemes.
- Optimized Storage Space: at any time, each user need only her own pairwise key and KEY in order to decrypt the MPEG stream.
- Support of dynamic join/leave/expel operations: at any time, any user can join the multicast connection. After the necessary authentication and the pairwise key is chosen, sender simply adds $\{KEY\}_{k_{n+1}}$ at the end of the key header, and the new user will be able to decrypt the data stream. (in case there is no change for KEY, sender only need to send $\{KEY\}_{k_{n+1}}$ to the new user). "Leave/expel" operations are also easy to support because sender can simply choose a new key KEY', and multicast it using the key header.
- Easy support of multiple-user/group join/leave/expel operations: there is no redundancy in handling multiple users.
- Timely Delivery: one obvious requirement is that all keys should be received by users before the decryption process. Because the key header is distributed together with the data stream, no additional synchronization enforcement is needed.

At the first look, this scheme seems expensive when only one or a few users leave or are expelled because the length of each header is, if each key is 128 bits, 16n bytes for a session with n users. But under some special circumstances, this scheme is actually an optimized algorithm. For example, if one half of the users in a multicast session are being expelled, the total bytes of transmitted new keys is 16*n/2=8n, comparing with Hierarchical Tree Approach which is at the level of 16*2n = 32n. As another example, if we apply a new KEY to each I frame of an MPEG stream, then n KEYs encrypted by each user's pairwise key have to be multicasted anyway (it is reasonable to assume that the leave/expel operation happens only at I frame), then we can claim that this scheme is optimized in terms of number of keys transmitted. This late observation is very important in our design of MPEG Multicasting Watermark Scheme described in the next section.

7.2.2 Handling Dynamically Changed Membership

Existing multicast security protocols in the open network environment are focused on the problem of group key distribution to all legitimate group members. Sending members can encrypt the multicast data with the group key, and receiving members can decrypt the multicast data with the group key. Because of the efficiency concern, most of the key management schemes use the multicast channel to distribute keys as well as data. But the multicast channel is prone to packet loss, and it is too costly to make the multicast channel reliable like TCP.

Most of the key distribution schemes fail in the presence of data loss. Consider the following scenarios:

1. Some senders did not get the key update message. They encrypt the data with the old key. Now expelled receivers can still decrypt the data.

2. Some receivers did not get the key update message. Expelled senders send the data using the old group key. These receivers can get confused.

3. Some receivers did not get the key update message. They will not be able to decrypt any data until the next key update message is received. Note that (3) may not be considered as a security violation, but it certainly lowers the service quality and is unacceptable in many

applications.

Our basic scheme described in the previous subsection also faces the same problems.

One way to solve the problems above is to use the session leader for the key update information delivery. Our basic scheme is modified as follows:

First, all group members establish a pairwise key with the session leader when the session begins or new member joins. Each user's public key is collected by the leader and distributed to all members.

Let uid be the member's user id ($suid$ to be sender's user id), $msgid$ be the message id, KEY_{msg} be the key that the sender generates on a per message (per frame) basis, K_{uid} be the pairwise key between the member uid and session leader (K_{suid} to be the pairwise key between the sender and session leader), Key_{suid}^{Pri} be the private key of sender uid, and Key_{sl}^{Pri} be the session leader's private key.

Second, the sender multicasts the following message:

$$suid, [msgid, [suid, KEY_{msg}]K_{suid}, [suid, data]KEY_{msg}]Key_{suid}^{Pri}$$

Session leader:

- Receives sender' message.
- Decrypts sender's message using sender's public key, and then applies key_{suid} to get KEY_{msg}.
- Verifies that the message is indeed from the sender as claimed.
- Verifies that the sender from the sending site is indeed a group member.
- Verifies that the suid and msgid are consistent in the sender's message.
- If suid is in the member list, then the session leader multicasts the following verification message:

  $$[suid, msgid, ACCEPT, [KEY_{msg}]key_{uid1}[KEY_{msg}]key_{uid2}...]Key_{sl}^{Pri}$$

- If suid is not in the member list, then the session leader multicasts the following rejection message:

  $[suid, msgid, REJECT]Key_{sl}^{Pri}$

Receivers:

- Receive a multicast message from sender. Get suid and msgid information.
- Buffer $msgid$ and $[suid, data]KEY_{msg}$ information.
- Receives session leader's verification message.
- Decrypt session leader's message: get [suid, msgid, KEY_{msg}] or (REJECT).
- Decrypt the sender's message with KEY_{msg}.

7.3 MPEG Multicasting Watermark Scheme

The copyright protection problem in multicast environment is much more difficult than the unicast and has not been addressed. On the one hand, it is desirable that each user gets different version of watermarked data and has her own unique copy. On the other hand, the goal of multicast scheme is to avoid transmitting multiple copies of data. These two aims look like contradictions to each other.

In this section, we address the above problem and design the scheme which targets the MPEG multicast and provides good solution to balance the trade-off between uniqueness and the sharing property of multicasting.

7.3.1 Basic Scheme

The basic idea to provide copyright protection for MPEG multicasting is to create two watermarked MPEG stream, assign a unique random 0-1 bit sequence to each user, and use this bit sequence to arbitrate between those two watermarked streams. Before the algorithm is described, we need to emphasize several issues:

1. Non-invertible watermarking scheme should be used, otherwise the watermarked MPEG video can be easily compromised.

2. Only I frames need to be watermarked. This is because, without corresponding I frames, the quality of P or B frames is bad enough and lack of commercial value even though some P or B frames may contain a lot of self decodable I-blocks. In the following discussions, we omit the presence of all P and B frames.

3. We may apply different watermarks to each I frame, or we can apply same watermark to all I frames in the MPEG video stream. For the simplicity, we choose the later one. Certainly, in order to create two watermarked streams based on the same MPEG video, two different watermarks have to be created and applied respectively.

The detailed scheme is as following:

1. Create two watermarks W_1 and W_2 using non-invertible watermarking scheme.

2. Denote each I frames in an MPEG stream, $I_1, I_2, ..., I_n$.

3. Watermark the MPEG stream using W_1 and W_2 individually, denote the resulting streams as $I_1 + W_0, I_2 + W_0, ..., I_n + W_0$, and $I_1 + W_1, I_2 + W_1, ..., I_n + W_1$, respectively.

4. Create random bit sequence for each user, the length of this bit sequence equals to the number of I frames in the MPEG stream and, in case of live video, the length of this bit sequence can be infinity.

5. For i-th (i=1, ..., num_of_I) watermarked I frame in stream j (j=0,1), we use a different key KEY_{ij} to encrypt it. Then we give either KEY_{i0} or KEY_{i1} to user n based on the random bit sequence of user n, i.e., if i-th bit of the random sequence is 0, give user n KEY_{i0}, otherwise, give user n KEY_{i1}.

6. The modified key header for i-th I frame looks like:

$$KEY_{i0K_1}KEY_{i1K_2}KEY_{i1K_3}...KEY_{i0K_n}$$

assume i-th bit of the random bit sequence is 0 for user 1, 1 for user 2, 1 for user 3, ..., 0 for user n.

7. The complete multicasted i-th I frame has the structure:

$$KEY_{i0K_1}KEY_{i1K_2}KEY_{i1K_3}...KEY_{i0K_n}\ I_i + W_{0KEY_{i0}}I_i + W_{1KEY_{i1}}$$

There are several comments about the basic scheme. First, due to a new key is used for each I frame, join/leave/expel operations are very easy to handle. This scheme can also easily allow suspension of certain members' access rights and resume later without requiring rejoin/re-authentication (to suspend a user n, simply does not give the new key to user n; to resume the access right of user n, simply give user n the new key). Second, although it seems that the scheme does not scale in key distribution, it is an optimized solution due to the frequently changed keys (as we discussed in section 7.2.1). On the other hand, the overhead to transmit rekey message is reasonable for middle size MPEG video multicasting, e.g., 1000 users. Assume each KEY is 128 bits or 16 bytes, then the length of key header is 16000 bytes and it is about the size of an I frame. The total bytes of transmitted data for an I frame is 16*n+2*I_Frame_Size. Compare to the unicast with watermark protection of n users, which needs to transmit n*I_Frame_Size bytes, the overhead is very small. The saving can be calculated as following:

$$1 - \frac{16*n+2*I\_Frame\_Size}{n*I\_Frame\_Size}$$

When n is very large, and assume the size of an I frame is 16K, then saving is about 99.9%. The larger the I frame is, the more saving this scheme can achieve.

7.3.2 Collusion Problem

By applying the basic MPEG Multicasting Watermark Scheme described in previous section, the "bad guy" who illegally distributes the video clips can be easily identified if there exists only one such attacker. However, collusion will be a concern. If two or more users collaborate, it is possible for them to generate a new sequence which is not created by the sender. For example, if User 1 has sequence 101100..., User 2 has sequence 110001..., they can easily create a sequence like 110001... and although we know 110001... is not a legal sequence, we can not identify who is responsible.

In general, this problem is too difficult to solve. We will consider special cases and provide solutions for them.

One special case is when no more than two users collaborate. If the bit sequence is long enough, eventually we can identify at least one such collaborator. Let us look at the following example to see how to identify the collaborators.

EXAMPLE 5 Let us assume there are four users in a multicast session, namely, A, B, C, and D. Let us also assume each user is assigned a unique bit sequence as follows:

```
A:  1 1 1 0 0 0
B:  1 0 0 1 1 0
C:  0 1 0 1 0 1
D:  0 0 1 0 1 1
```

Now if an illegal sequence "1 1 0 0 1 0" was found, how can we find out who are the collaborators? Because our assumption is that only two of them are "bad guys", then there are totally six possible combinations: AB, AC, AD, BC, BD, CD. Bit 1 of the illegal sequence is 1, then we can eliminate CD because if C and D collaborate the only possible number is 0 at Bit 1. Similarly, Bit 2 eliminates BD, Bit 3 eliminates AD, Bit 4 eliminates BC, Bit 5 eliminates AC, and Bit 6 eliminates CD again. The only combination left is AB, therefore, we can identify A and B to be the collaborators.

But sometimes it is impossible to identify both collaborators. For instance, let us assume the illegal sequence is "1 0 1 0 0 0". Using the same sequences assigned to A, B, C, and D in EXAMPLE 5, we find out that, Bit 1 eliminates CD, Bit 2 eliminates AC, Bit 3 eliminates BC, Bit 4 eliminates BC (again!), Bit 5 eliminates BD, and Bit 6 eliminates CD (again!). AB and AD are two possible combinations left and we can not decide which one is the collaborated pair. However, we notice that both AB and AD have A, therefore, we can conclude that A must be one of the combinations. This is the reason we claim that we can "identify at least one such collaborator" but not both collaborators.

A more secure solution is that, at certain I frame, we generate n different versions of watermarked I frame. There is a trade-off between how many such I frames we choose and the security.

In summary, the collusion problem is still an open problem.

7.4 Notation

In this section, we summaries the notations used in this chapter.

I: the original video frame, particularly means MPEG I frame.

W: the watermark.

uid: the multicasting session member's user id.

$suid$: the sender's user id, sender is also a session member.

$msgid$: the message id.

KEY_{msg}: the key to encrypt the message (video clip data). Each sender generates this on a per message (per frame) basis.

K_{uid}: the pairwise key between the member uid and session leader.

K_{suid}: the pairwise key between the sender and session leader.

Key_{suid}^{Pri}: the private key of sender uid.

Key_{sl}^{Pri}: the session leader's private key.

7.5 Conclusion

In this chapter, we present a secure multicast protocol with copyright protection targeting the multimedia data. It is simple, does not require any security mechanism on the network switches or routers, and can be applied to any existing multicast architectures. It is robust in the presence of data loss and it can handle dynamic session membership changes. It is efficient in reasonable bandwidth requirement and timely key delivery. It also provides the copyright protection in the multicast environment.

In general, the security and copyright protection have the nature that excludes other users and multicast tries to share information as much as possible. These two issues contradict to each other. This makes the security and copyright protection in multicast environment very difficult. There are still many open problems need to be solved in the future, for example, the collusion problem in using watermark in multicast.

Chapter 8

Research Accomplishments and Future Work

8.1 Contributions

The major research accomplishments of our current multimedia security framework are summarized below.

- We analyzed the statistical behavior of MPEG video streams.
- We proposed a new MPEG encryption algorithm - VEA which utilizes the special MPEG property. VEA is fast, secure and preserves the compressed stream size after encryption..
- We provided and analyzed attacks on existing MPEG encryption algorithms.
- We introduced the idea of using standard encryption function to generate the watermark.
- We proposed new watermark algorithms and proved their non-invertibility.
- We presented and analyzed non-invertible watermarking schemes for image, video, and audio. Experimental results were also presented.
- We are the first to identify the customer's right problem and we designed the protocols to solve the problem.
- We discussed security issues within multicast environment and provide simple, efficient, and secure multicast protocols with copyright protection for the multimedia data.

8.2 Future Research

With the booming of the Internet, more and more multimedia data become available, therefore, to provide the multimedia security and copyright protection services is more urgent. It is still interesting and challenge to create an integrated system to provide those services. A concrete example of such system is a security gateway or firewall with those security and copyright protection functionalities.

Such secure gateway for a company is very useful because (1) when a multimedia material is created in a company, the company owns the material, not the individual user, hence, it makes sense to provide the ownership watermarking protection at the multimedia-secure gateway which has the company's watermark and can include it properly into the material, (2) performance of individual computing nodes increases because they do not have to use up CPU cycles for security algorithms, (3) the multimedia secure gateway is only used when material is sent/received out/from outside of the company, hence the overall performance of the intra-network within the company improves as the communication within the company does not have to carry additional security-related information, (4) in general, there is no need to protect the copyright inside a company (group), therefore we only need to enforce security service for incoming and outgoing multimedia data, (5) not every individuals in a company (group) can be trusted (or always be trusted), a security gateway is needed for intentional/unintentional information leaking and copyright violation.

Another challenge work is the collusion problem in using watermark in multicasting environment. As we pointed out in the thesis, the security and copyright protection have the nature that excludes other users and multicast tries to share information as much as possible. These two head to opposite directions. This makes the security and copyright protection in multicast environment very difficult.

Bibliography

[1] M. Abadi and R. Needham. Prudent Engineering Practice for Cryptographic Protocols. In *Proceedings of the 1994 IEEE Symposium on Security and Privacy*, pages 122–136, Los Alamitos, CA, 1994.

[2] I. Agi and L. Gong. An Empirical Study of Mpeg Video Transmissions. In *Proceedings of the Internet Society Symposium on Network and Distributed System Security*, pages 137–144, San Diego, CA, February 1996.

[3] N. Asokan, M. Schunter, and M. Waidner. Optimistic Protocols for Fair Exchange. In *Proceedings of 4th ACM Conference on Computer and Communications Security*, Zurich, April 1997.

[4] A. Ballardie. Core Based Trees (CBT) Multicast Architecture. Ietf draft, February 1996.

[5] A. Ballardie. Scalable Multicast Key Distribution. Rfc1949, May 1996.

[6] T. Ballardie and J. Crowcroft. Multicast-specific Security Threats and Counter-measures. In *Proceedings of the Symposium on Network and Distributed System Security*, San Diego, CA, February 1995.

[7] T. Ballardie, P. Francis, and J. Crowcroft. Core Based Trees: An Architecture for Scalable Inter-domain Multicast Routing. In *Proceedings of the ACM SIGCOMM '93*, San Francisco, CA, September 1993.

[8] D. Bayer, S. Haber, and W.S. Stornetta. Improving the Efficiency and Reliability of Digital Timestamping. In *Sequences II: Methods in Communication, Security, and Computer Science*, pages 329–334. Springer-Verlag, 1993.

[9] M. Bellare, J. Garay, R. Hauser, A. Herzberg, H. Krawczyk, M. Steiner, G. Tsudik, and M. Waidner. iKP – a Family of Secure Electronic Payment Protocols. In *Proceedings of the First USENIX Workshop on Electronic Commerce*, New York, July 1995.

[10] S. Bellovin and M. Merritt. Limitations of the Kerberos Authentication System. In *Proceedings of USENIX Conference*, pages 253–267, Winter 1991.

[11] W. Bender, D. Gruhl, and N. Morimoto. Techniques for Data Hiding. In *Proceedings of the SPIE Conference on Storage and Retrieval for Image and Video Databases III*, volume 2420, pages 164–173, San Jose, CA, February 1995.

[12] E. Biham and A. Shamir. Differential Cryptanalysis of DES-like Cryptosystems. In *Advances in Cryptology Crypto '90*, pages 2–21. Springer-Verlag, 1991.

[13] R. Bird, I. Gopal, A. Herzberg, P. Janson, S. Kutten, R. Molva, and M. Yung. Systematic Design of a Modular Family of Attack-resistant Authentication Protocols. *IEEE Journal on Selected Areas in Communications*, 11(5):679–693, June 1993.

[14] L. Boney, A. H. Tewfik, and K. N. Hamdy. Digital Watermarks for Audio Signals. In *Proceedings of 1996 IEEE International Conference on Multimedia Computing and Systems*, pages 473–480, Hiroshima, Japan, June 1996.

[15] J. Brassil, S. Low, N. Maxemchuk, and L. O'Gorman. Electronic Marking and Identification Techniques to Discourage Document Copying. In *Proceedings of IEEE INFOCOM'94*, volume 3, pages 1278–1287, Toronto, June 1994.

[16] M. Burrows, M. Abadi, and R. Needham. A Logic of Authentication. Technical Report 39, DES SRC, Palo Alto, CA, 1990.

[17] G. Caronni. Assuring Ownership Rights for Digital Images. In *Proceedings of Reliable IT Systems, VIS'95*. Vieweg Publishing Company, 1995.

[18] David Chaum. Achieving Electronic Privacy. *Scientific American*, pages 96–101, August 1992.

[19] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press and McGraw-Hill Book Company, New York, 1990.

[20] B. Cox, J. D. Tygar, and M. Sirbu. Netbill Security and Ttransaction Protocol. In *Proceedings of USENIX Workshop on Electronic Commerce*, July 1995.

[21] I. J. Cox, J. Kilian, T. Leighton, and T. Shamoon. Secure Spread Spectrum Watermarking for Multimedia. *IEEE Transaction on Image Processing*, 6(12):1673–1687, 1997.

[22] I. J. Cox and M. Miller. A Review of Watermarking and the Importance of Perceptual Modeling. In *Proceedings of the IS&T/SPIE Conference on Human Vision & Electronic Imaging II*, volume 3016, pages 92–99, San Jose, CA, February 1997.

[23] S. Craver, N. Memon, B. Yeo, and M. Yeung. Can Invisible Watermarks Resolve Rightful Ownerships? Technical Report RC 20509, IBM Research Division, July 1996.

[24] W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Trans. Inform. Theory*, IT-22:644–654, Nov. 1976.

[25] J. Dittmann, F. Nack, A. Steinmetz, and R. Steinmetz. Interactive Watermarking Environments. In *Proceedings of IEEE Multimedia Systems '98*, pages 286–294, Austin, TX, June 1998.

[26] J. Zhao E. Koch, J. Rindfrey. Copyright Protection for Multimedia Data. In *Proceedings of the International Conference on Digital Media and Electronic Publishing*, pages 203–213, London, UK, 1996.

[27] G. C. Langelaar et.al. Robust Labeling Methods for Copy Protection of Images. In *Proceedings of the SPIE Conference on Storage and Retrieval for Image and Video Databases V*, volume 3022, pages 298–309, San Jose, CA, February 1997.

[28] Jean-Paul Boly et.al. The Esprit Project Cafe - High Security Digital Payment Systems. In *ESORICS '94, LNCS 875*, pages 217–230. Springer-Verlag, 1994.

[29] S. Deering et.al. The PIM Architecture for Wide-Area Multicast Routing. In *IEEE/ACM Transactions on Networking*, volume 4, April 1996.

[30] M. K. Franklin and M. K. Reiter. Fair Exchange with a Semi-trusted Third Party. In *Proceedings of The Fourth ACM Conference on Computer and Communications Security*, pages 1–6, Zurich, April 1997.

[31] L. Gong. Using One-way Functions for Authentication. *ACM Computer Communication Review*, 19(5):8–11, October 1989.

[32] L. Gong and P. Syverson. Fail-Stop Protocols: An Approach to Designing Secure Protocols. In *Proceedings of 5th International Working Conference on Dependable Computing for Critical Applications*, pages 44–55, Urbana-Champaign, IL, September 1995.

[33] S. Haber and W.S. Stornetta. How to Timestamp a Digital Document. *Journal of Cryptology*, 3(2):99–111, 1991.

[34] F. H. Hartung and B. Girod. Watermarking of MPEG-2 Encoded Video without Decoding and Reencoding. In *Proceedings of the SPIE Conference on Multimedia Computing and Networking 1997*, volume 3020, pages 264–274, San Jose, CA, February 1997.

[35] X. Lai, J. L. Massey, and S. Murphy. Markov Ciphers and Differential Cryptanalysis. In *Advances in Cryptology – EUROCRYPT '91*, pages 17–38. Springer-Verlag, 1992.

[36] Y. Li, Z. Chen, S. Tan, and R. Campbell. Security enhanced mpeg player. In *Proceedings of IEEE First International Workshop on Multimedia Software Development (MMSD'96)*, Berlin, Germany, March 1996.

[37] S. H. Low, N. F. Maxemchuk, and S. Paul. Anonymous Credit Cards. In *Proceedings of the 2nd ACM Conference on Computer and Communication Security*, 1994.

[38] B. M. Macq and J. Quisquater. Cryptology for Digital TV Broadcasting. In *Proceedings of the IEEE*, pages 944–957, June 1995.

[39] T.B. Maples and G.A. Spanos. Performace Study of a Selective Encryption Scheme for the Security of Networked, Real-time Video. In *Proceedings of 4th International Conference on Computer Communications and Networks*, Las Vegas, Nevada, September 1995.

[40] G. Medvinsky and B. C. Neuman. Electronic Currency for the Internet. *Electronic Markets*, 3(9-10):23–24, October 1993.

[41] G. Medvinsky and B. C. Neuman. Netcash: A Design for Practical Electronic Currency on the Internet. In *Proceedings of the 1st ACM Conference on Computer and Communication Security*, November 1993.

[42] J. Meyer and F. Gadegast. Security Mechanisms for Multimedia Data with the Example mpeg-1 Video. Available on WWW via http://www.powerweb.de/phade/phade.html, 1995.

[43] S. Mittra. Iolus: A Framework for Scalable Secure Multicasting. In *Proceedings of ACM SIGCOMM '97*, Cannes, France, September 1997.

[44] J. Moy. Multicast Extension to OSPF. Rfc1584, March 1994.

[45] R. M. Needham and M. D. Schroeder. Using Encryption for Authentication in Large Networks of Computers. *Communications of the ACM*, 21(12):993–999, December 1978.

[46] NIST. Data Encryption Standard. FIPS Publication 46-2, 1993.

[47] International Standards Organization. Information technology – Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 mbit/s – Part 3: Audio. International Standard ISO/IEC IS 11172-3, 1993.

[48] D. Pan. A Tutorial on Mpeg/audio Compression. *IEEE Multimedia*, pages 60–74, Summer 1995.

[49] L. Qiao and K. Nahrstedt. Watermarking Method for MPEG Encoded Video: Towards Resolving Rightful Ownership. Technical Report UIUCDCS-R-97-2032, University of Illinois at Urbana-Champaign, Urbana, IL, 1997.

[50] M. K. Reiter and S. G. Stubblebine. Toward Acceptable Metrics of Authenticatio. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, pages 10–20, Oakland, CA, May 1997.

[51] R. L. Rivest and RSA Laboratories. Faq 3.0 on Cryptography: What is Rc4? Available on WWW via http://www.rsa.com/rsalabs/newfaq/q87.html, 1998.

[52] M. Schneider and S.-F. Chang. A Robust Content Based Digital Signature for Image Authentication. In *Proceedings of the 1996 IEEE International Conference on Image Processing*, Lausanne, Switzerland, Sept 1996.

[53] R. L. Rivest A. Shamir and L. M. Adleman. A Method for Obtaining Digital Signatures and Public-key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[54] G. Simmons. *Contemporary Cryptology The Science of Information Integrity*. IEEE Press, 1992.

[55] M. Sirbu and D. Tygar. Netbill: An Internet Commerce System Optimized for Network-delivered Services. *IEEE Personal Communications*, 2(4):34–39, August 1995.

[56] J. R. Smith and B. O. Comiskey. Modulation and Information Hiding in Images. In *Workshop on Information Hiding*, Isaac Newton Institute, University of Cambridge, UK, May 1996. Springer-Verlag Lecture Notes in Computer Science Volume 1174.

[57] J. G. Steiner, C. Neuman, and J. I. Schiller. Kerberos: An Authentication Service for Open Network Systems. In *Proceedings of USENIX Conference*, pages 191–202, Winter 1988.

[58] P. Syverson. Limitations on Design Principles for Public Key Protocols. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 62–73, Oakland, CA, May 1996.

[59] P. Syverson and C. Meadows. A Logical Language for Specifying Cryptographic Protocol Requirements. In *Proceedings of the 1993 IEEE Computer Society Symposium on Research in Security and Privacy*, Los Alamitos, CA, 1993. IEEE Computer Society Press.

[60] P. Syverson and C. Meadows. Formal Requirements for Key Distribution Protocols. In *A. D. Santis, editor, Advances in Cryptology – EUROCRYPT'94*, Berlin, 1995. Springer-Verlag Lecture Notes in Computer Science Volume 950.

[61] K. Tanaka, Y. Nakamura, and K. Matsui. Embedding Secret Information into a Dithered Multi-level Image. In *Proceedings of 1990 IEEE Military Communications Conference*, pages 216–220, 1990.

[62] L. Tang. Methods for Encrypting and Decrypting MPEG Video Data Efficiently. In *Proceedings of The Fourth ACM International Multimedia Conference (ACM Multimedia'96)*, pages 219–230, Boston, MA, November 1996.

[63] L. F. Turner. Digital Data Security System. Patent IPN WO 89/08915, 1989.

[64] R. G. van Schyndel, A. Z. Tirkel, and C. F. Osborne. A Digital Watermark. In *Proceedings of the International Conference on Image Processing*, volume 2, pages 86–90, IEEE, 1994.

[65] D. Waitzman, S. Deering, and C. Partridge. Distance vector multicast routing protocol. Rfc1075, November 1988.

[66] D. Wallner, E. Harder, and R. Agee. Key Management for Multicast: Issues and Architectures. Ietf draft, JULY 1997.

[67] R. B. Wolfgang and E. J. Delp. A Watermarking Technique for Digital Imagery: Further Studies. In *Proceedings of the International Conference on Imaging Science, Systems, and Technology*, Las Vegas, Nevada, JULY 1997.

[68] K. Yamamoto. An Integration of PGP and Mime. In *Proceedings of the Internet Society Symposium on Network and Distributed System Security*, pages 17–24, San Diego, CA, February 1996.

[69] J. Zhou and D. Gollmann. A Fair Non-repudiation Protocol. In *Proceedings of 1996 IEEE Symposium on Security and Privacy*, pages 55–61, Oakland, CA, May 1996.

[70] P. Zimmermann. *The Official PGP User's Guide*. MIT Press, 1995.

Vita

Lintian Qiao was born on July 21, 1962 in Beijing, China. He received his B.S. degree in Mathematics from Peking University in 1984. From 1984 through 1988 he was a software engineer in Beijing Institute of Computer Technology where he worked on designing the Information System of Chinese Ministry of Agriculture and the office management system for Beijing City Hall.

In 1988, he enrolled as a graduate student at the Marquette University, Milwaukee, Wisconsin. He received his M.S. in Mathematics and Computer Science in 1990. From 1990 through 1993, he worked as a senior programmer/analyst at TASC, Inc. in Chicago, Illinois.

From 1993 through 1998, he was a Ph.D. student in the Department of Computer Science at the University of Illinois at Urbana-Champaign. From 1995 through 1998, he was a research assistant in Professor Klara Nahrstedt's Multimedia Operating Systems and Networking (MONET) Research Group. After finishing his Ph.D. in 1998, he joined Lucent in Naperville, Illinois, where he worked in the Broadband Data Group to develop features for packetized voice over ATM/IP network.