Contributions to image encryption and authentication

Takeyuki Uehara
University of Wollongong

Contributions to Image Encryption and Authentication

A thesis submitted in partial fulfillment of the
requirements for the award of the degree

Doctor of Philosophy

from

UNIVERSITY OF WOLLONGONG

by

Takeyuki Uehara

Department of Computer Science
October 2003

Declaration

This is to certify that the work reported in this thesis was done by the author, unless specified otherwise, and that no part of it has been submitted in a thesis to any other university or similar institution.

Takeyuki Uehara
October 21, 2003

Abstract

Advanced digital technologies have made multimedia data widely available. As multimedia applications become common in practice, security of multimedia data has become main concern. Digital images are widely used in various applications, that include military, legal and medical systems and these applications need to control access to images and provide the means to verify integrity of images.

Image encryption algorithms protect data against unauthorized access. In almost all cases image data is compressed before it is stored or transmitted because of the enormity of multimedia data and their high level of redundancy. Compressing plaintext before applying the encryption algorithm effectively increases security of the overall system. However direct application of encryption algorithms to image data *i*) requires high computational power and *ii*) introduces delay in real-time communication. If a data compression algorithm can be made to also provide security, less processing overhead could be expected as a single algorithm achieves two goals.

Image authentication provides the means to verify the genuineness of images. *Authentication codes* provide a method of ensuring integrity of data. The challenge in image authentication is that in many cases images need to be compressed and so the authentication algorithms need to be compression tolerant. Cryptographic authentication systems are sensitive to bit changes and so are not suitable for image authentication.

In this thesis, we study existing image encryption and authentication systems and demonstrate various attacks against these systems. We propose a JPEG encryption system that encrypts only part of the data, and a JPEG2000 encryption system that uses a simple operation, i.e. permutation, and show methods to minimize the computation cost for encryption. We also propose an image authentication system that remains tolerant to changes due to JPEG lossy compression.

Acknowledgments

I would like to thank my supervisor Dr. Rei Safavi-Naini and Dr. Philip Ogonbuna for guiding and encouraging me throughout this project. I would also like to thank Dr. Wanqing Li and Dr. Xing Zhang for their interest in this project. I would also like to thank my colleagues, Gareth Charles Beatt Brisbane, Chandrapal Kailasanathan, Dr. Nicholas Sheppard, Angela Piper, Vu Dong To, Qiong Liu, the people in Centre for Computer Security Research (CCSR) and Dr. John Fulcher. The work of the author is partially supported by Motorola Australian Research Centre (MARC).

Publications

The results of research in this thesis were published as follows.

- Takeyuki Uehara and Reihaneh Safavi-Naini, *Chosen DCT Coefficients Attack on MPEG Encryption Schemes*, Proc. of IEEE Pacific-Rim Conference on Multimedia, 316-319, 2000

- Takeyuki Uehara and Reihaneh Safavi-Naini and Philip Ogunbona, *Securing Wavelet Compression with Random Permutations*, Proc. of IEEE Pacific-Rim Conference on Multimedia, 332-335, 2000

- Takeyuki Uehara and Reihaneh Safavi-Naini, *On (In)security of "A Robust Image Authentication Method"*, Proc. of IEEE Pacific-Rim Conference on Multimedia (PCM 2002),1025-1032, 2002

- Takeyuki Uehara, Reihaneh Safavi-Naini and Philip Ogunbona, *A Secure and Flexible Authentication System for Digital Images*, ACM Multimedia Systems Journal to appear, 2003

Patent applications are as follows.

- JPEG2000 encryption system

  Takeyuki Uehara (University of Wollongong), Reihaneh Safavi-Naini (University of Wollongong), Philip Ogunbona (Motorola Australian Research Centre) and Motorola

- JPEG encryption system

  Takeyuki Uehara (University of Wollongong), Reihaneh Safavi-Naini (University of Wollongong), Philip Ogunbona (Motorola Australian Research Centre) and Motorola

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Advanced digital technologies have made multimedia data available on large capacity storage devices such as hard disk, CD, and DVD, and through high speed networks. As multimedia applications become common in practice, the security of multimedia data has become a main concern.

Image encryption algorithms protect data against unauthorized access. Encryption is used in applications such as subscribed digital TV broadcasting [63] which require the data to be hidden from an unsubscribed person, and Digital Versatile Disc (DVD) [100].

Wide use of images in digital form and the case of malicious modification of digital data has raised the need for image authentication. Digital images in legal and medical applications and also news reporting [67] require proof of authenticity of images and an assurance that the image has not been modified. To provide digital images usable in many applications, it is essential to provide various security measures for images.

1.1 Motivation

Digital Images

Different types of data have different degrees of sensitivity to change. For example, the executable code of a computer program may not tolerate a single bit change because it can result in the program crashing or producing different result. Image data can tolerate a higher level of change because the limited sensitivity of human eyes leaves small changes undetectable. The information which cannot be sensed by human eyes is considered irrelevant and is often removed using *lossy compression*.

In almost all cases image data is compressed before it is stored or transmitted because of *i*) the enormous size of multimedia data files and *ii*) the very high redundancy in the data and so incorporating security in the compression system is a very attractive

approach.

There are several standards for image and motion picture compression. The *Joint Photographic Experts Group (JPEG)* standard [45] is one of the most widely used standards for still images. JPEG specifies the de-compression algorithm and representations of compressed data. It also provides guidelines for implementation of the algorithm. JPEG uses two different classes of compression algorithms. That is, *lossy* and *lossless* compression. In JPEG, the lossy compression uses the Discrete Cosine Transform (DCT) [3]. For the entropy coding, Huffman or arithmetic coding is used to compress (and decompress) the quantized DCT coefficients. A new image compression standard, JPEG2000 [43], uses the Discrete Wavelet Transform [21, 64] and provides various improvements over the JPEG system. The *MPEG (Moving Pictures Experts Group)* standard by the ISO/IEC [42] is a compression algorithm for multimedia data including video and audio data. It provides the standards not only for compressing video and audio streams, but also the meta-data of other types of data such as text. In MPEG, video and audio streams are independent from each other. For video compression, MPEG compresses a sequence of images by removing the *spatial redundancy* using the DCT for the transformation followed by quantization and entropy coding, and *temporal redundancy* using block-based motion compensated prediction (MCP) [23].

Image Encryption

Encryption algorithms protect data against unauthorized access. With the rapid growth of the Internet, controlling access to data is of increasing importance and hence encryption is of much wider use. Digital images are no exception. Compressing plaintext before applying the encryption algorithm effectively increases the security of the overall system [96]. However direct application of encryption algorithms to image data *i*) requires high computational power and *ii*) introduces delay in real-time communication.

For example, the encryption algorithm *AES (Advanced Encryption Standard)*[73] which is known to be fast, requires about 15 cycles per byte for encryption and decryption [5, 61]. Let us consider an MPEG decoder [74], which uses a RISC (Reduced Instruction Set Computer) CPU core running on about 150 MHz. Decryption of an 8 M bps MPEG stream, which is the common bit rate for DVD video, requires 15 mega cycles (100 milliseconds for 150 MHz clock) and so the decryption increases the decoding time by 100 milliseconds for 1 second video data, that is, 10 % of the time to

play video is spent for decryption, which is too expensive. In the hardware implementation case, adding encryption algorithms to the encoder and the decoder increases the complexity of circuits and results in an increase in the cost of manufacturing.

If a data compression algorithm can be made to also provide security, less processing overhead could be expected as a single algorithm achieves two goals. The combination of image compression and encryption will only be successful if the resulting system *i*) does not considerably reduce compression rate, *ii*) requires less processing time than compression followed by encryption and *iii*) can provide demonstrable security. Many existing systems satisfy only one or two of the three requirements which shows that designing a successful system is a difficult task.

Images may be partially encrypted. For secure image encryption, it must be difficult to recover the image, or a perceptually equivalent version of the image, where perceptual equivalence may be defined depending on the application. For example, all images that are the result of compression down to certain quality level followed by decompression, may be considered perceptually equivalent. An encrypted image should mask the visual information of an image and make the content visually incomprehensible. The security is measured in terms of the difficulty of recovering same visual information from the encrypted form.

Some applications may not require strong security. For example in consumer products, DVD's Content Scrambling System (CSS) was invented in 1996 [34]. The system used 40 bit key encryption which was considered weak but provided sufficient security for preventing an average user with limited knowledge and resources to illegally copy the content. We consider a system *reasonable secure* if the system is secure against such a user. (CSS was broken in 1999 [46]: the weakness of the algorithm allowed plaintext to be recovered with a cost equivalent to 2^{25} which is much lower than an exhaustive key search with a cost of 2^{40} [100, 101].)

Image Authentication

Image authentication provides the means to verify the authenticity of images. *Message authentication codes* and digital signatures are the main cryptographic primitives to provide data integrity [103].

Image and video data are displayed on a range of displays with different resolutions, and processed on a range of devices with various computing capability, from a high-end workstation to a handheld device. To display the visual information on different platforms, different versions which are encoded according to the requirements of the platform, will be used. However, all such versions must pass the verification test on an

authenticator that is calculated from the original data.

For example, a digital camera may authenticate pictures taken by the camera and output the compressed image together with the authenticator. The user of the camera may wish to keep the high quality original and then re-compress the images to send to others. In this case the re-compressed images have to pass the verification test applied to the re-compressed images and the authenticator generated by the camera.

There is a set of operations commonly applied on images that could be acceptable as not changing the semantic content although they will change the values of pixels. For example, enhancing images improves contrast and picture clarity but can be considered as an acceptable change of the image. In many applications, it is important that image authentication tolerates modifications made by such acceptable operations.

Cryptographic authentication systems are sensitive to bit changes and so are not suitable for image authentication in the scenarios described above.

1.2 Objective

Our objectives are to investigate compression-encryption schemes and compression tolerant authentication schemes applied to digital images. The questions addressed in this research are *i*) whether it is possible to achieve efficiency, measured in terms of level of security, compression rate and processing time, by integrating image compression and encryption, and *ii*) if there are methods for image authentication which are compression tolerant.

1.3 Contributions

In this thesis firstly we present new attacks on existing image encryption systems and then show methods of improving the security of these systems.

1.3.1 Image Encryption

There are two approaches of combining compression and encryption.

Elementary cryptographic operations : Using less expensive elementary cryptographic operations such as random permutation (or *transposition*), data can be effectively hidden. Since these operations are simple, encryption does not have a high computation cost. The challenge is how to achieve reasonable security with such simple operations.

Selective encryption : It is possible to reduce the computational cost by reducing the size of the data to be encrypted.

By hiding important parts of the data or crucial parameters of the compression algorithm, the data stream can be protected. For effective protection, these parts must be carefully chosen so that the encrypted stream cannot be decoded without these parts, or even if it is decoded, the quality of the obtained image is highly degraded.

We demonstrate attacks to find the secret key of encryption systems in both approaches. We also show methods which incorporate encryption into JPEG and JPEG2000 compression systems.

1.3.2 Image Authentication

We demonstrate an attack on a compression tolerant image authentication system and then show methods of protection against the attack. We also give a secure and flexible compression tolerant image authentication scheme that provides various levels of security.

1.3.3 Organization of Thesis

The thesis is organized as follows. In Chapter 2, we give an overview of information theory, data compression, security and image compression systems. In Chapter 3, first we review combined data compression and encryption schemes and image encryption systems, and then image authentication systems. In Chapter 4, we demonstrate new attacks against the JPEG and MPEG image encryption systems. In Chapter 5, we propose a new JPEG encryption system using selective encryption. In Chapter 6, first we propose a new combined image encryption and wavelet-based compression system that can provide various degrees of security using random permutations, and then we extend the method to be applied to JPEG2000 image compression system. In Chapter 7, we show cryptanalysis of SARI [58, 59] image authentication system and demonstrate a new attack against the system, and then we propose a new image authentication system which has a number of advantages over SARI. In Chapter 8 we summarize our results and conclude the thesis.

1.4 Images

Gray scale images and color images used in the experiments in this thesis are shown in Table 1.1 and Figure 1.1, and in Table 1.2 and Figure 1.2, respectively.

Table 1.1: The sizes of gray scale images.

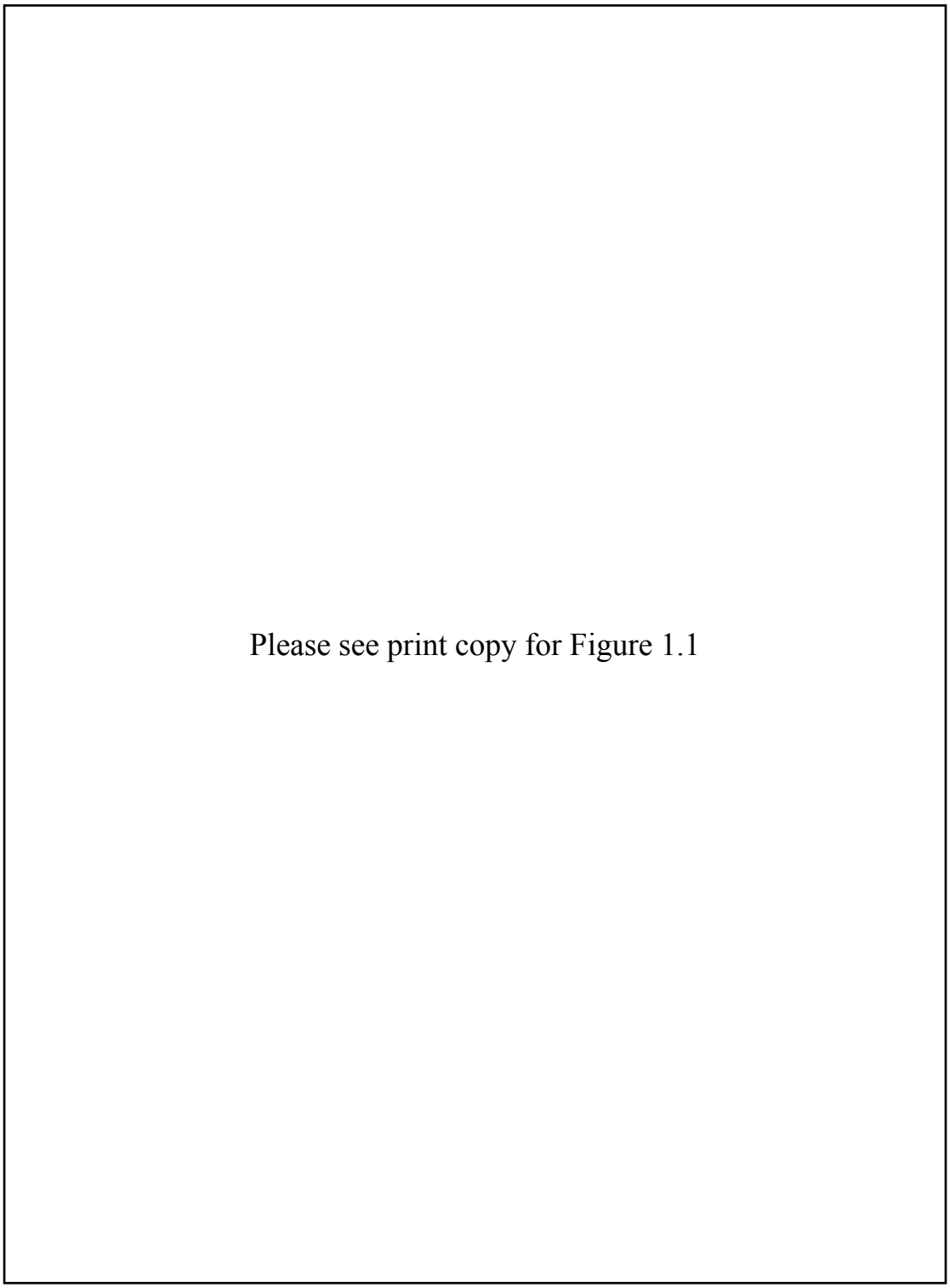
Files	image size (pixels)	pixel value
`airfield.pgm`	256×256	8 bits
`airplane.pgm`	512×512	8 bits
`lena.pgm`	512×512	8 bits
`mandoril.pgm`	512×512	8 bits
`peppers.pgm`	512×512	8 bits

Table 1.2: The sizes of color images.

Files	image size (pixels)	pixel values
`lena.ppm`	512×512	24 bits (8 bits $\times$ 3 for RGB)
`mandoril.ppm`	512×512	24 bits (8 bits $\times$ 3 for RGB)
`peppers.ppm`	512×512	24 bits (8 bits $\times$ 3 for RGB)

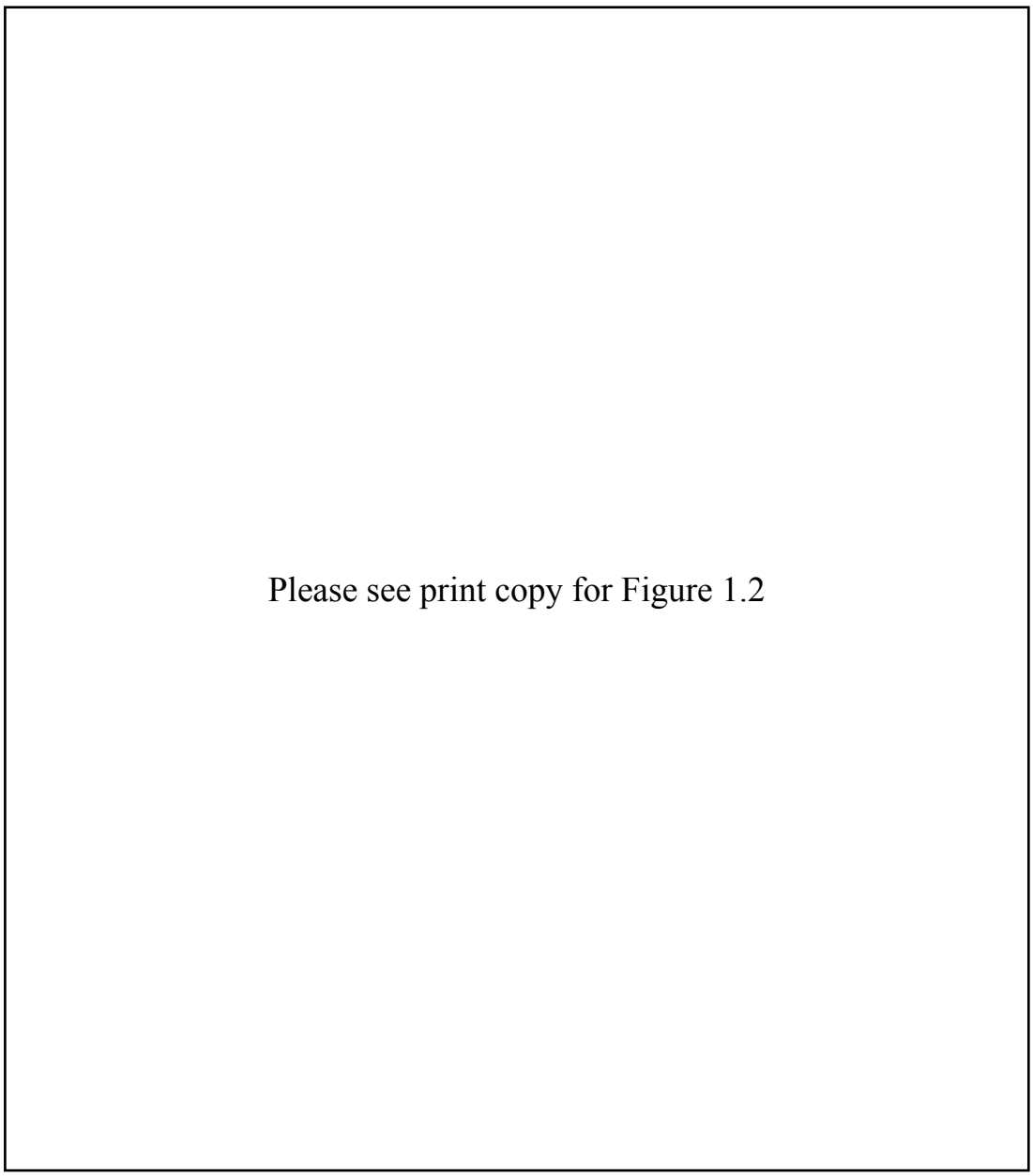
1.5 Notations

MSB	the most significant bit.
LSB	the least significant bit.
MSBs	the most significant bits.
LSBs	the least significant bits.
${}^{n}\mathrm{P}_{r}$	$\frac{n!}{(n-r)!}$.
${}^{n}\mathrm{C}_{r}$	$\frac{n!}{(n-r)!r!}$.
$rint()$	an integer rounding function.
$Q^{(u,v)}$	quantization step of DCT coefficient at (u,v) position in JPEG compression.
${F_p}^{(u,v)}$	DCT coefficient at (u,v) position in block p in JPEG compression.
${T_p}^{(u,v)}$	quantized DCT coefficient at (u,v) position in block p in JPEG compression.
$\tilde{F}_p^{(u,v)}$	dequantized DCT coefficient at (u,v) position in block p in JPEG compression.
${F_{MAX}}^{(u,v)}$	the maximum value of a DCT coefficient in JPEG compression
${F_{MIN}}^{(u,v)}$	the minimum value of a DCT coefficient in JPEG compression
${Y_g}^{(u,v)}$	the sum of linear combination of DCT coefficients of group g at (u,v) position (image authentication system).
$\tilde{Y}_g^{(u,v)}$	the sum of linear combination of dequantized DCT coefficients of group g at (u,v) position (image authentication system).
${A_1}^{(u,v)}$	linear combination coefficient for DCT coefficient at (u,v) position(image authentication system).
${A_{MAX}}^{(u,v)}$	the maximum value of linear combination coefficient for DCT coefficient at (u,v) position(image authentication system).
${A_{MIN}}^{(u,v)}$	the maximum value of linear combination coefficient for DCT coefficient at (u,v) position(image authentication system).



(e)

Figure 1.1: Gray scale images : (a) `airfield.pgm`, (b) `airplane.pgm`, (c) `lena.pgm`, (d) `mandoril.pgm`, and (e) `peppers.pgm`.



(c)

Figure 1.2: Color images : (a) `lena.ppm`, (b) `mandoril.ppm`, and (c) `peppers.ppm`.

Chapter 2

Background

2.1 Introduction

In this chapter we briefly review theories, constructions and systems that will be used in the rest of the thesis. Firstly, we introduce information theory and the basics of data compression and security systems. Next, we examine image compression algorithms, and outline several standard compression systems.

2.2 Information Theory

Communication System

A *communication system*, Fig. 2.1, consists of a *message source*, an *encoder*, a *channel*, and a *decoder*. The message source produces messages to be transmitted. The encoder performs *source coding*, which converts the messages into a form suitable for transmission. The channel is the medium through which the encoded messages are transmitted. The noise may interfere with the communication over the channel. The decoder recovers the encoded messages (possibly with some information loss) for the receiver of the information.

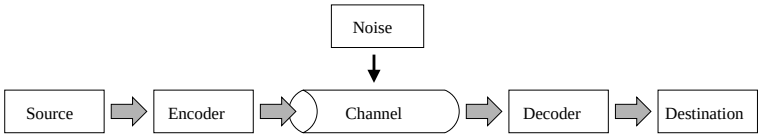


Figure 2.1: Communication system.

Uncertainty and Entropy

Information is related to *uncertainty*. In a communication system, a message is transmitted from an information source. A *discrete source* consists of an alphabet set X

together with a probability distribution p on the set. A message consists of a sequence of symbols, chosen from the alphabet set according to the distribution. Once a symbol is emitted, the uncertainty about that symbol is removed.

The message source can be modeled with a discrete random variable. The average amount of information in the source alphabet can be measured using the *entropy function* defined as follows [8]. Let X denote the discrete random variable that takes values x_i, $1 \leq i \leq M$, $i \in \mathcal{Z}$ with probabilities $p(x_i) > 0$. Then the average information per source symbol is given by

$$H_M(X) = -\sum_{i=1}^{M} p(x_i) \log p(x_i) \ .$$

This is called the *entropy* of the random variable.

$H(X)$ can be interpreted as the average amount of information obtained after observing one element of X. It can also be interpreted as the average uncertainty about an element of X. For example, assume that in an experiment a fair die is rolled. Each number appears with equal chance and so the probability of a number from 1 to 6 is $p(X = 1) = p(X = 2) = p(X = 3) = p(X = 4) = p(X = 5) = p(X = 6) = \frac{1}{6}$. Then $H(X)$ is given by $H(X) = -\sum_{i=1}^{6} \frac{1}{6} \log_2 \frac{1}{6} = -log_2 \frac{1}{6} \approx 2.58$ bits. Before rolling the die, the average uncertainty over the experiment is 2.58 bits. After the outcome of the experiment is known, 2.58 bits of information is obtained and the uncertainty is removed.

For a pair of discrete random variables $X \in \{x_1, x_2, ..., x_M\}$ and $Y \in \{y_1, y_2, ..., y_L\}$, with joint probability distribution $p(x_i, y_j) > 0$, $1 \leq i \leq M$ and $1 \leq j \leq L$, the joint entropy of X and Y is given by

$$H(X, Y) = -\sum_{i=1}^{M} \sum_{j=1}^{L} p(x_i, y_j) \log p(x_i, y_j) \ .$$

Joint entropy satisfies the condition $H(X, Y) \leq H(X) + H(Y)$ and with equality if and only if X and Y are independent. The conditional entropy of X and Y is given by

$$H(Y|X) = -\sum_{i=1}^{M} p(x_i) \sum_{j=1}^{L} p(y_j|x_i) \log p(y_j|x_i)$$

and satisfies the condition $H(X|Y) \leq H(X)$ with equality if and only if X and Y are independent.

2.3 Data Compression

Data compression is an outgrowth of information theory. The aim of data compression is to find a short description for messages of a source.

For a fixed channel, compressing messages results in a more efficient use of the channel. There are two types of compression algorithms : *lossless compression* and *lossy compression*. In this section, we look at the lossless compressions and the lossy compressions are reviewed in Section 2.5. In lossless compression systems the compressed data can be used to recover an exact replica of the source output. In lossy compression only an approximate form of the source output can be recovered.

2.3.1 Source Coding

The information source may be represented by a continuous or a discrete random variable. We only consider discrete sources. In *source coding* the encoder encodes a symbol produced by the source into a *codeword* over an alphabet. A codeword is a string over the code alphabet. The codeword length may be fixed or variable. An example of a *fixed length code* is the ASCII code. Morse code is an example of a *variable length code*. By assigning shorter codewords to more frequent source symbols, a shorter average length and hence a more efficient code is obtained.

2.3.2 Optimal Codes

Let the source be represented by a discrete random variable, X, that takes value from a set $\{x_1, x_2, ..., x_M\}$. In source coding each symbol is encoded into a codeword which is a sequence of symbols over another alphabet set, $A = \{a_1, a_2, ..., a_D\}$. Let the length of the codeword x_i be l_i. The average length of a code is given by

$$\bar{l} = \sum_{i=1}^{M} p(x_i) l_i \ . \tag{2.1}$$

The most efficient code is the one with the minimum average length. An important property of a code is that it should be *uniquely decodable*. A code is uniquely decodable if any encoded string corresponds to a unique source string. A *prefix code* is a code in which no codeword is a prefix of any other codeword. It has the property that it is uniquely decodable. The *Kraft Inequality* gives a necessary and sufficient condition that must be satisfied by a prefix code [51].

For any prefix code over an alphabet A, the Kraft inequality is :

$$\sum_{i=1}^{M} D^{-l_i} \leq 1 \ . \tag{2.2}$$

The relationship between the entropy and the average codeword length is

$$H(X) \leq \bar{l} \log D \ . \tag{2.3}$$

Equality holds if and only if $p(x_i) = D^{-l_i}, \forall i$; in this case, $H(X) = \bar{l} \log D$. There exist D-ary (alphabet consisting of D symbols) prefix codes which satisfy

$$\frac{H(X)}{\log D} \leq \bar{l} \leq \frac{H(X)}{\log D} + 1 \ . \tag{2.4}$$

The prefix code with the shortest average length is called the *optimal* prefix code.

2.3.3 Constructions of Optimal Codes

Huffman Code

A method of constructing an optimal prefix code is given by Huffman [35]. Assume the set of symbols $X = \{x_1, x_2, ..., x_M\}$ with probabilities $p(x_1), p(x_2), ..., p(x_M)$, where $p(x_1) \geq p(x_2) \geq ... \geq p(x_M)$, is given. Assume there are D symbols in the code alphabet. Then the algorithm is as follows:

1. Combine D symbols with the smallest probabilities to construct a symbol $x_{M-D+1,...,M}$ with probability $\sum_{i=M-D+1}^{M} p(x_i)$, and replace $x_{M-D+1}, ..., x_M$ by the new symbol. This reduces the size of the source by $D-1$. Repeat the procedure until the number of remaining symbols becomes D.

2. Assign a single symbol from the code alphabet to each resulting symbol. This is the code for the reduced source.

3. For any symbol obtained from combining D symbols, construct D codewords by appending a code symbol to the codeword assigned to the combined symbol. Repeat the procedure until all original symbols have assigned codewords.

An example is shown in Fig 2.2. The symbol set is $X = \{1, 2, 3, 4, 5\}$ with probabilities $0.25, 0.25, 0.2, 0.15, 0.15$ respectively and $D = 2$.

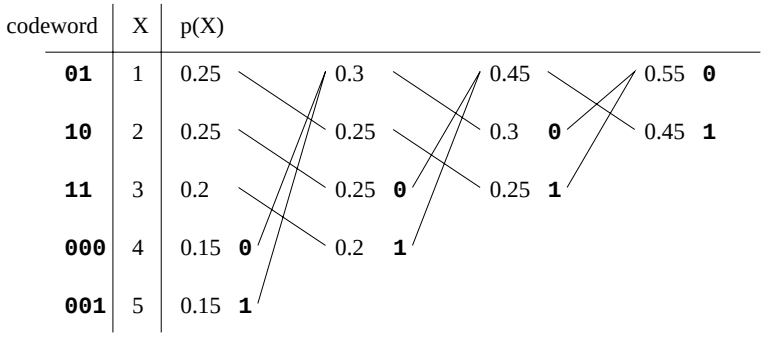


Figure 2.2: Example of a Huffman code.

Shannon-Fano-Elias Code

Shannon-Fano-Elias coding encodes source symbols using a cumulative distribution. It is not an optimal coding algorithm but forms the basis of arithmetic coding, to be discussed in Section 2.3.3, which is an optimal coding algorithm. Let X be a set of source symbols, where $X = \{1, 2, ..., M\}$ and the probabilities be $p(X) = \{p(1), p(2), ..., p(M)\}$ where $p(x) > 0$ for all x. Then the cumulative distribution function $F(X)$ is defined as $F(x) = \sum_{i=1}^{x} p(i)$. We define a function $\bar{F}(X)$ as $\bar{F}(x) = \sum_{i=1}^{x-1} p(i) + \frac{1}{2}p(x)$. If we use a binary representation of $\bar{F}(x)$ after the decimal point, i.e. removing 0., with the length $l(x) = \lceil \log \frac{1}{p(x)} \rceil$, we can construct a uniquely decodable code. Shannon-Fano-Elias coding assigns an integral number of bits to each codewords and in this way it differs from arithmetic coding, which will be discussed later.

Ziv-Lempel Coding

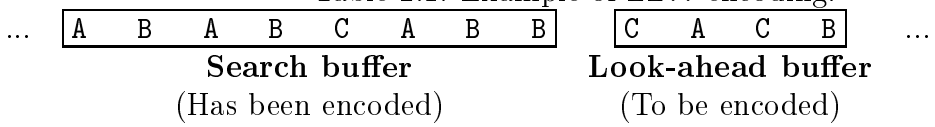
Two *dictionary based compression* methods are given by Ziv and Lempel [130, 131].

LZ77

The first Ziv-Lempel coding method [130], called *LZ77*, uses a *sliding window* to the input stream. The window consists of two parts, *i)search buffer*, which includes the symbols which have already been encoded, and *ii)look-ahead buffer*, which contains the symbols which will be encoded.

The following example shows the method of encoding. Assuming that the encoder encodes the input stream from left to right, the sliding window, the search buffer and the look-ahead buffer contain 16, 8, and 4 symbols, respectively. (Typically the size of the search buffer is the order of kilobytes and that of the look-ahead buffer is tens of bytes.) In the example of Table 2.1, the string "`ABABCABB`" has already been encoded and the encoder is going to encode "`CACB...`".

Table 2.1: Example of LZ77 encoding.



1. The encoder scans the search buffer from right to left, looking for the string "CACB...".

2. If any matches are found, the encoder chooses the longest and left-most match. The output of the encoder is *i*) the position (the distance from the look-ahead buffer) of the matched string, *ii*) the length of the matched string, and *iii*) the unmatched symbol in the string. If no match is found, the encoder outputs $(0, 0, X)$ where X is the first unmatched symbol in the look-ahead buffer.

   In this example, "CA" is found at position 4 and so the output is $(4, 2, "C")$.

3. Then the sliding window is shifted to the right direction by the amount which is equal to the matched string size + 1.

The above procedure is repeated and as a result, variable length symbols produce the fixed sized code.

LZ78

The second method [131], called *LZ78*, partitions a message into variable-length blocks and constructs a dictionary for it. Let a message, $X = (x_1, x_2, ..., x_n)$, be over an M symbol alphabet. Then the first entry in the dictionary is $B_1 = (x_1)$. The shortest prefix $B_2 = (x_2, ..., x_i)$ of the sequence $(x_2, ...x_n)$ that is not in the dictionary is added to the dictionary and the procedure is repeated. Each entry in the dictionary is referred to by a pair of integers (j, x_k) in such a way that x_k is the last symbol in B_l and B_j is the sequence obtained by removing x_k from B_l. The codewords are given from the pair of integers by $Mj + x_k$. Ziv-Lempel code is a *universal source coding*, which compresses data without prior knowledge of the source distribution.

Data Compression Models

Statistical compression systems such as Huffman and arithmetic coding can be divided into two parts: a *model* part that models the source and a *coder* part which uses the information given by the model to encode the incoming symbols. If a model is fixed throughout the coding of the message, it is a *static model* and the system is a *non-adaptive data compression system*. In an *adaptive data compression system*, the model

is updated by the incoming data to reflect its local statistics.

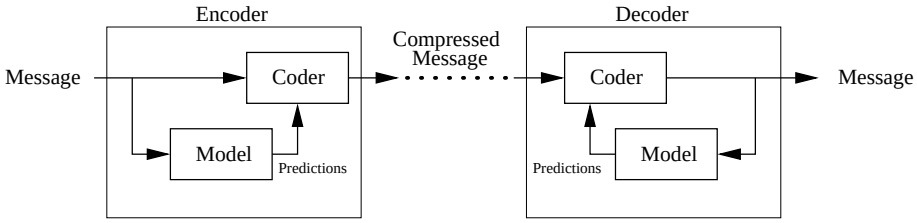


Figure 2.3: Data compression model.

In adaptive statistical compression algorithms such as adaptive Huffman [50] or adaptive arithmetic coding [121], the probabilities of source symbols and the codewords assigned to the source symbols are updated according to the incoming source messages. The encoder can update the distribution of symbols by observing input symbols and the decoder can follow the change of the encoder by observing the decoded symbols.

Arithmetic Coding

Arithmetic coding is an optimal data compression algorithm [121, 117]. It was discovered independently by Pasco [77] and Rissanen [83].

Arithmetic coding encodes a message into a bit string which represents a real number interval within the interval $[0, 1)$. The encoder starts with an initial interval, usually $[0, 1)$, and narrows it as new symbols arrive such that the amount of narrowing is determined by the probability of the incoming symbol.

When a message consists of a sequence of m symbols, $\psi_1, \psi_2, ...\psi_m$, the required length to encode the message is

$$\sum_{i=1}^{m} -log_2 p(\psi_i) \tag{2.5}$$

where $p(\psi_i)$ is the probability of the symbol ψ_i.

The compressed data consists of an integral number of bits. In arithmetic coding, a whole message is represented by an integral number of bits but each symbol does not have such a restriction. If the result of encoding a symbol includes a fraction of a bit, the fraction is passed to the next symbol. This is the advantage of arithmetic coding over Huffman coding. In case of Huffman coding, each symbol should be translated into an integral number of bits and so the fraction of a bit is rounded up if there is any. The extra fraction for each output symbol cumulates as encoding proceeds and hence adds to the length of the encoded message.

2.4 Security Systems

2.4.1 Symmetric Key Encryption

A *symmetric key cryptosystem* allows secure communication over an insecure channel between two parties who share a key. A *symmetric encryption algorithm* is a collection $\varepsilon = \{E_K \ : K = 1, 2 \cdots N\}$ of invertible transformations indexed by a piece of information called *key*.

To encrypt a *plaintext* message X, the transmitter, who shares a key k with the receiver, finds the *ciphertext* $Y = E_k(X)$ and sends it to the receiver who can use the inverse transformation to recover X. The attacker does not know the key. An attacker may use an *exhaustive key search* strategy to determine the key and in this case the number of keys gives an upper bound on the security of the system.

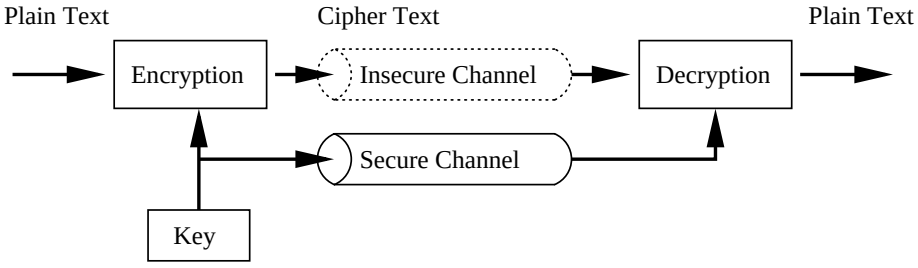


Figure 2.4: Model of symmetric encryption system.

Commonly used symmetric key encryption algorithms include DES [38] and AES [73].

2.4.2 Public Key Cryptography

A *public key cryptosystem* allows secure communication over an insecure channel between two parties using a pair of a *public key* and a *private key* [68]. The main difference between a public key cryptosystem and a symmetric cryptosystem is that only the private key needs to be secret and the public key can be made publicly accessible in the public key cryptosystem through an authentic channel.

A *public key encryption algorithm* is a collection of pairs of encryption and decryption transformations $\phi = \{(E_e, D_d) \ : e, d \in K = 1, 2 \cdots N\}$, where a decryption transformation D_d is the inverse of the associated encryption transformation E_e.

To encrypt a plaintext message X, the transmitter receives receiver's public key e over an insecure channel, finds the *ciphertext* $Y = E_e(X)$ and sends it to the receiver who can use the inverse transformation $X = D_d(Y)$ to recover X. For the security of a

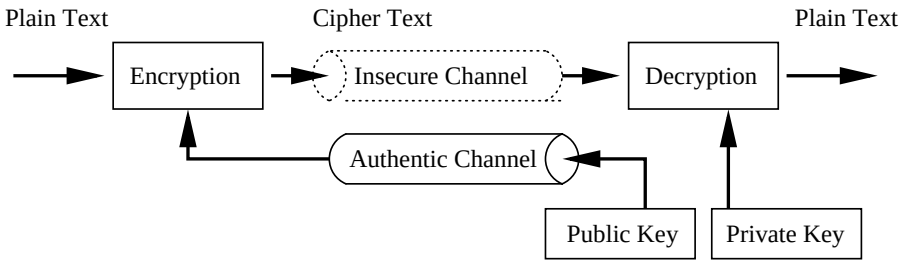


Figure 2.5: Model of public key encryption system.

public key cryptosystem, given e it must be infeasible to determine the corresponding d. A common public key cryptosystem is RSA [87].

2.4.3 Authentication

Authentication includes *i*) *Entity authentication (entity identification)*: to guarantee that entities are those who they claim to be, *ii*) *Data authentication (data integrity)*: to guarantee the integrity of information, that is, the information has not been manipulated by unauthorized parties, *iii*) *Data origin authentication (message authentication)*: to assure that the entity is the original source of data and the data has not been tampered with, *iv*) *Key authentication*: to assure the identity of party which share a secret key, and *v*) *Non-repudiation*: to prevent an entity from denying previous commitments or actions [68].

Authentication systems are used to provide entity authentication, message authentication, data authentication, non-repudiation and key authentication. *Digital signature schemes* are asymmetric key primitives for authentication. *Message authentication codes* are symmetric key primitive that are used for message authentication.

2.4.4 Digital Signature

A *digital signature* is the main primitive for *authentication*, providing non-repudiation through binding the identity of an entity to the signed document [68]. A basic *digital signature system* δ is a *signature generation algorithm* S_A and a *signature verification algorithm* V_A and so $\delta = (S_A, V_A)$. A signature generation S_A for an entity A is a mapping $S_A : X \rightarrow S$, where X is the message set and S is the signature set. A signature verification V_A is a mapping given as $V_A : X \times S \rightarrow \{true, false\}$.

A signer A computes $s = S_A(X)$ and transmits the pair (X, s) to a verifier, who computes $u = V_A(m, s)$.

For example, in the digital signature system using a public key system, the signing

transformation consists of the generation of the hash of the message $h(X)$ using a cryptographic hash function and the calculation of $s = D_d(h)$ using A's private key d. Assuming that Alice calculates $s = S_{Alice}(X)$ and transmits the pair (X, s) to Bob but Bob receives (X', s'), then the verification algorithm produces,

$$u = \begin{cases} true, \ if \ E_e(s') = h(X') \\ false, \ if \ E_e(s') \neq h(X') \end{cases}$$

where e is Alice's public key. There are standards for digital signature algorithms such as ANSI X9.31 [39] and Digital Signature Standard [76].

2.4.5 Message Authentication Codes

Message authentication codes (MAC) provide assurances of the source of a message and its integrity [68]. They are *keyed hash functions* which have two input parameters, a message X and the secret key k, given as $Y = h_k(X)$. The difference between a message authentication code and a digital signature scheme is that the digital signature is publicly verifiable but the MAC is verifiable only by the receiver who shares the secret information k.

2.4.6 Attacks against Encryption Systems

An encryption system can be attacked by an enemy. The goal of the attacker may be *i*) key recovery, or *ii*) finding plaintext.

There are several attack models as shown below.

Ciphertext-only attack An attacker knows only the ciphertext. That is, he/she has access to $E_k(X)$. This is possible if he/she can eavesdrop the channel.

Known plaintext attack An attacker knows a number of pairs of plaintexts and their corresponding ciphertexts and tries to discover a key used for generation of ciphertexts, or a plaintext corresponding to a ciphertext which is not in the known set. This attack scenario is possible if he/she can eavesdrop the channel and has partial access to the plaintexts.

Chosen plaintext attack An attacker can choose one or more plaintexts and can obtain the corresponding ciphertexts. This is possible if he/she has access to the encryption system.

Adaptive chosen plaintext attack A chosen plaintext attack where a plaintext can be chosen depending on previously obtained ciphertexts.

Chosen ciphertext attack An attacker can choose one or more ciphertexts and observe the corresponding plaintexts. This is possible if he/she has access to the decryption system.

Adaptive chosen ciphertext attack A chosen ciphertext attack where a ciphertext can be chosen depending on previously observed plaintexts.

2.4.7 Attacks against Authentication Systems

Against authentication systems, the attacker may attempt impersonation, substitution, repudiation, and finding the key of the MAC [103]. For example, suppose the impersonation and the substitution in a digital signature scheme are as follows. Let X be a message and its signature be Y. Then in impersonation, the attacker, Oscar, creates his own message X and then chooses its signature Y so that the receiver, Bob, identifies Y as signed by another entity, Alice. In substitution, Oscar intercepts (X, Y) generated by Alice, and modifies it to (X', Y') or he creates (X', Y') and hopes that it remains acceptable.

Examples of the attacks against hash are as follows.

1. *Guessing attack*:

   For a message X with n-bit hash $h(X)$, choose a random bit-string X' of the length $l < a$ where a is a constant and check if X' satisfies $h(X) = h(X')$. The probability of $h(X) = h(X')$ is 2^{-n}.

2. *Birthday attack*:

   Let X and Y be the legitimate and fraudulent message, respectively, and $h(X)$ be an n bit one-way hash function.

   (a) The attacker generates $t = 2^{n/2}$ messages X_i,$1 \leq i \leq t$, each by making small modification on X, and computes $h(X_i)$ and store the results.

   (b) Loop: he generates Y_i and computes $h(Y_i)$ and search X_i such that $h(Y_i) = h(X_i)$,$1 \leq i \leq t$.

   A match can be expected after t trials.

2.4.8 Redundancy of a Language

A natural language can be seen as a message source and so can be analyzed using information theory [8]. Let S be the alphabet of a language with M symbols, and S^k denote a string of k characters. Then the *rate of language* r_k for messages of length k is the average amount of information per character in these messages and is given by

$$r_k = \frac{H_2(S^k)}{k} \ . \tag{2.6}$$

The *absolute rate of a language* is the maximum amount of information that could be encoded in each character using the alphabet S assuming that all combinations of symbols are equally likely. It is given by

$$R = \log_2 M \ . \tag{2.7}$$

The *redundancy of a language* D with rate r is given by

$$D = R - r \ . \tag{2.8}$$

For English, r_k has been estimated to be 1.0 to 1.5 bits/letter and R is 4.7 bits/letter [68]. The high value of R means that the English language is highly *redundant*. For a language, less redundancy means more statistical independence of the successive characters in a message. Redundancy of languages can be removed by using data compression algorithms.

2.4.9 Unicity Distance

When a language is redundant, the knowledge of statistical properties of the language can be used to attack an encryption system [8]. The *unicity distance* U_Δ is the minimum amount of ciphertext required by the attacker using ciphertext attacks with unlimited computational resources to uniquely identify the key [68], and is given by

$$U_\Delta = \frac{H(K)}{R} \tag{2.9}$$

where $H(K)$ is the entropy of keys and R is the redundancy of the plaintext. When the unicity distance is small, a short ciphertext gives enough information to uniquely identify the key K and hence the security is weak. By lowering R, that is compressing the source and reducing redundancy, the unicity distance is increased.

2.4.10 Data Compression and Security

As shown in Section 2.4.9, compressing a message source before encryption increases security by removing redundancy of the source and increasing the unicity distance. An encryption algorithm produces ciphertexts that look like a random sequence and so do not have any apparent redundancy. This means that ciphertexts do not compress much. Encryption is now widely used in computer systems, and so compression before encryption is an important strategy for efficient use of resources. Combining compression and encryption algorithm has the advantage of added efficiency and automatic compression before encryption.

2.5 Image Compression

Image compression is essential for multimedia applications since the volume of image data is very large and so can be costly for transmission and storage without compression. The digitization procedure of an image consists of *sampling* and *quantization* [88]. A digital image is commonly represented by a rectangular array of dots called *pixels* (still images) or *pels* (fax and video images). The number of bits assigned to a pixel value determines the number of colors, or shades in *gray scale* images, a pixel can show. The reduction of data size is achieved by either *lossless* or *lossy* compression. In the lossless case, the compression removes the redundancy in the image data. In the lossy case, the compression removes the irrelevant information in the image data, that is, the information which is less important in terms of *Human Visual System (HVS)*. The most commonly used quality measure for lossy compression is the *peak signal to noise ratio (PSNR)*. Let the pixels of the original and the reconstructed image be P_i and Q_i, $i \in \{1, 2, ..., n\}$, respectively. Then the PSNR is defined as

$$PSNR = 20 \log_{10} \frac{\max_i |P_i|}{RMSE}$$

where $RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (P_i - Q_i)^2}$ (RMSE: *root mean square error*). There are different approaches to both lossless and lossy compression. For example, JPEG and JPEG 2000 support both lossless and lossy mode. Lossy compression is used for the images commonly seen on Internet and the lossless compression is often used for medical images.

2.5.1 Transform

It is known that the neighboring pixels of a digitized image are highly correlated [88]. In image compression, a transform first decorrelates the pixel data and then the transformed data is quantized and entropy-coded. In the following sections, we review two transforms, the *discrete cosine transform* and *discrete wavelet transform*, which are the most widely used in image compression.

Discrete Cosine Transform

The *discrete cosine transform* (DCT) [3] is a signal analysis tool that can be used to decompose an image signal into its frequency components. The energy compaction efficiency of the DCT is known to be near-optimal for the first order autoregressive model, AR(1), that best models image data and so is widely used in the decomposition stage of natural image compression systems [104]. For example, it is the transform of choice in JPEG [45] and MPEG2 [70] coding standards. For efficient computation of *transform coefficients*, the image is partitioned into blocks of sub-images and the transform is applied to each block independently. From the coding efficiency and fast computation viewpoints the commonly used block size is 8×8; this is the block size used in JPEG and MPEG2 coding standards. The transform coefficients can be classified into two groups, namely, *DC and AC coefficients*. The DC coefficient is the mean of the pixel values of the image block and carries most of the energy in the block. The AC coefficients carry energy depending on the amount of detail in the image block. However, usually most of the energy is compacted in the DC coefficient and a few AC coefficients. Image compression systems exploit the energy compaction property of the DCT and use quantization to produce a more compact representation of the image because of the small amount of energy in the higher frequency AC coefficients.

Let the pixel values, x_{ij} in a given $N \times N$ image block be represented as the matrix $[X]$. Let $[A]$ denote the matrix of DCT basis vectors given by $a_{ij} = \cos((j-1)\frac{(2i-1)\pi}{16})$, $i, j \in \{1, 2, ..., 8\}$. Then $[A]$ is given by

1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
0.980785	0.831470	0.555570	0.195090	-0.195090	-0.555570	-0.831470	-0.980785
0.923880	0.382683	-0.382683	-0.923880	-0.923880	-0.382683	0.382683	0.923880
0.831470	-0.195090	-0.980785	-0.555570	0.555570	0.980785	0.195090	-0.831470
0.707107	-0.707107	-0.707107	0.707107	0.707107	-0.707107	-0.707107	0.707107
0.555570	-0.980785	0.195090	0.831470	-0.831470	-0.195090	0.980785	-0.555570
0.382683	-0.923880	0.923880	-0.382683	-0.382683	0.923880	-0.923880	0.382683
0.195090	-0.555570	0.831470	-0.980785	0.980785	-0.831470	0.555570	-0.195090

and the DCT of the data is given as the matrix $[C]^{DCT}$,

$$[C]^{DCT} = [A][X][A]^t \ . \tag{2.10}$$

The image block is recovered through the inverse transform as,

$$[X']^{IDCT} = [A][C][A]^t \ . \tag{2.11}$$

For a given image block k, the pixel values can be written as a decomposition into a DC and an AC component. Thus,

$$[X]_k = [X]_k^{DC} + [X]_k^{AC} \ . \tag{2.12}$$

Of course, $[X]_k^{DC}$ is constant over all $i, j \in N$ and its components are given by,

$$(x^{DC})_k^{(i,j)} = \frac{1}{N^2} F_k^{(1,1)} \tag{2.13}$$

where $F_k^{(1,1)}$ is a DC coefficient. In the case of JPEG where the block size is 8×8 the multiplying factor is $\frac{1}{64}$.

Discrete Wavelet Transform

The wavelet transform decomposes a signal into different scale levels, called *multiresolution signal decomposition* [64]. If this is applied to an image, the image is decomposed into the signals which contain different level of details, and so different levels of importance. This property enables the allocation of bandwidth to different scale levels according to the importance of the signals for efficient image coding [4, 118].

The discrete wavelet transform consists of a sequence of wavelet filter banks, each of which consist of a low-pass and a high-pass filter. The filter bank decomposes the signal into coarse and detailed parts.

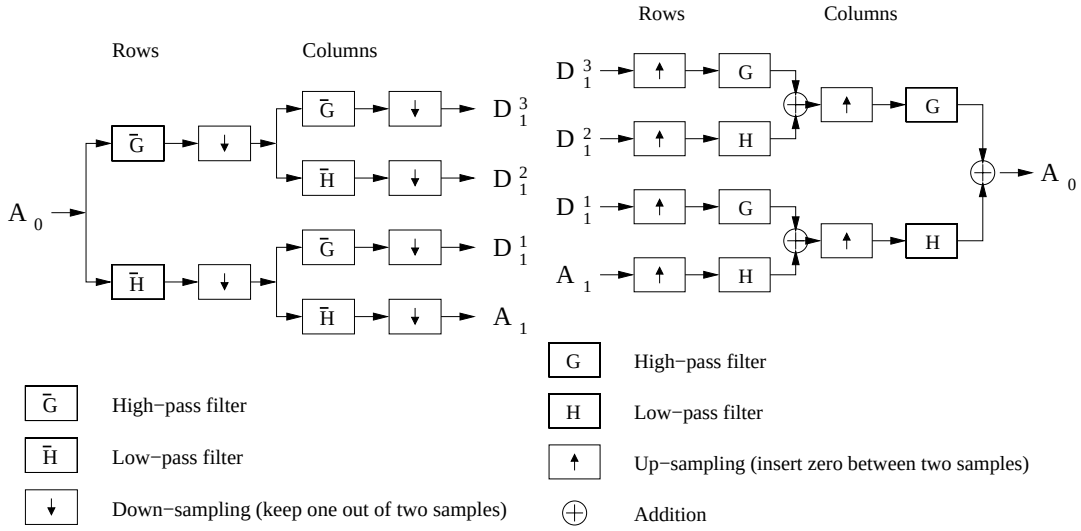


Figure 2.6: Wavelet transform (left) and its inverse transform (right).

In the 2-dimensional transform, firstly each row of the image is decomposed into coarse and detailed parts and down-sampled, and then each column of the coarse and detailed parts is decomposed into coarse and detailed parts and is down-sampled again. This results in four parts: coarse-coarse, coarse-detailed (both from the row coarse part), detailed-coarse and detailed-detailed parts (both from row detailed part). The last three parts compose the output subbands while the coarse-coarse part is the input of the next filter bank. Hence each filter bank produces three subbands and four subbands for the last filter bank.

In an implementation, the high-pass and low-pass filter is represented by a set of *filter coefficients* and filtering the input uses convolution of the filter coefficients with the input data. Let s_i be a one-dimensional signal and c_k denote filter coefficients where $k \in \{0, 1, ..., M\}$, then the convolution is

$$s'_i = \sum_{k=0}^{M} c_k s_i$$

Let a hi-pass and low-pass filter coefficients be $c_{k1}^{(H)}$ and $c_{k2}^{(L)}$, respectively, where $k1 \in \{0, 1, ..., M_1\}$ and $k2 \in \{0, 1, ..., M_2\}$. Then the output of the hi-pass and low-pass filters $s_i^{(H)}$ and $s_i^{(L)}$ is obtained by $s_i^{(H)} = \sum_{k=0}^{M_1} c_k^{(H)} s_{i-k}$ and $s_i^{(L)} = \sum_{k=0}^{M_2} c_k^{(L)} s_{i-k}$, respectively.

There are various wavelet filters such as by Daubechies [21] and by Antonini *et al.* [4]. For example, the filter coefficients of the Daubechies (9,7) filter used in JPEG2000 [44] are (0.026749, −0.016864, −0.078223, 0.266864, 0.602949, 0.266864, −0.078223, −0.016864, 0.026749) and (0.091272, −0.057543, −0.591272, 1.115087, −0.591272,

−0.057543, 0.091272) for the high-pass and low-pass filters, respectively.

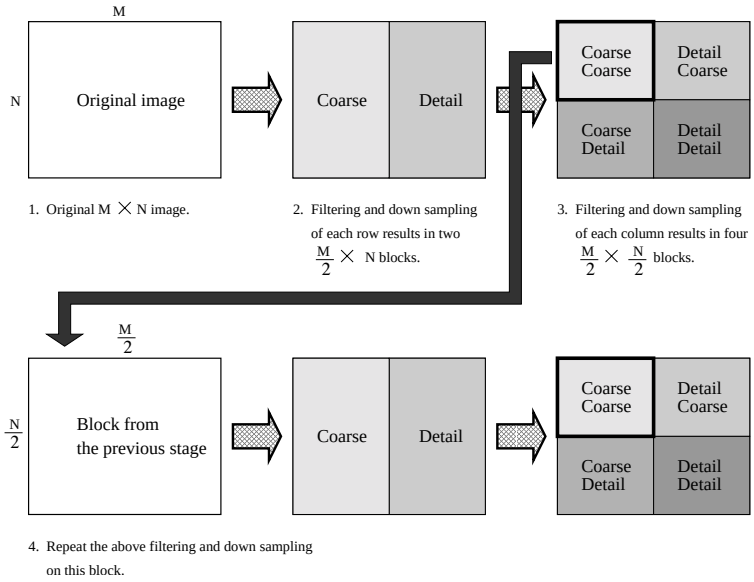


Figure 2.7: Wavelet decomposition of an image.

2.5.2 Quantization

In general, transformed coefficients of an image are real numbers and due to the redundancy of the data, the number of coefficients can be large but many of them can have close values [88]. Quantization converts the real numbers to a set of integers. The conversion is not invertible. By quantization, if real number coefficients are converted into a small number of integers, the data can be compressed.

There are various quantization methods. First we review *scalar quantization* which is used in JPEG compression system and then more complex *vector quantization*.

Scalar Quantization

Scalar quantization converts a real number into an integer [88]. The method used in JPEG is *uniform scalar quantization* [45]. Using uniform scalar quantization, a real number coefficient is divided by an integer quantization step and the result is integer rounded. More details are shown in Section 2.5.3.

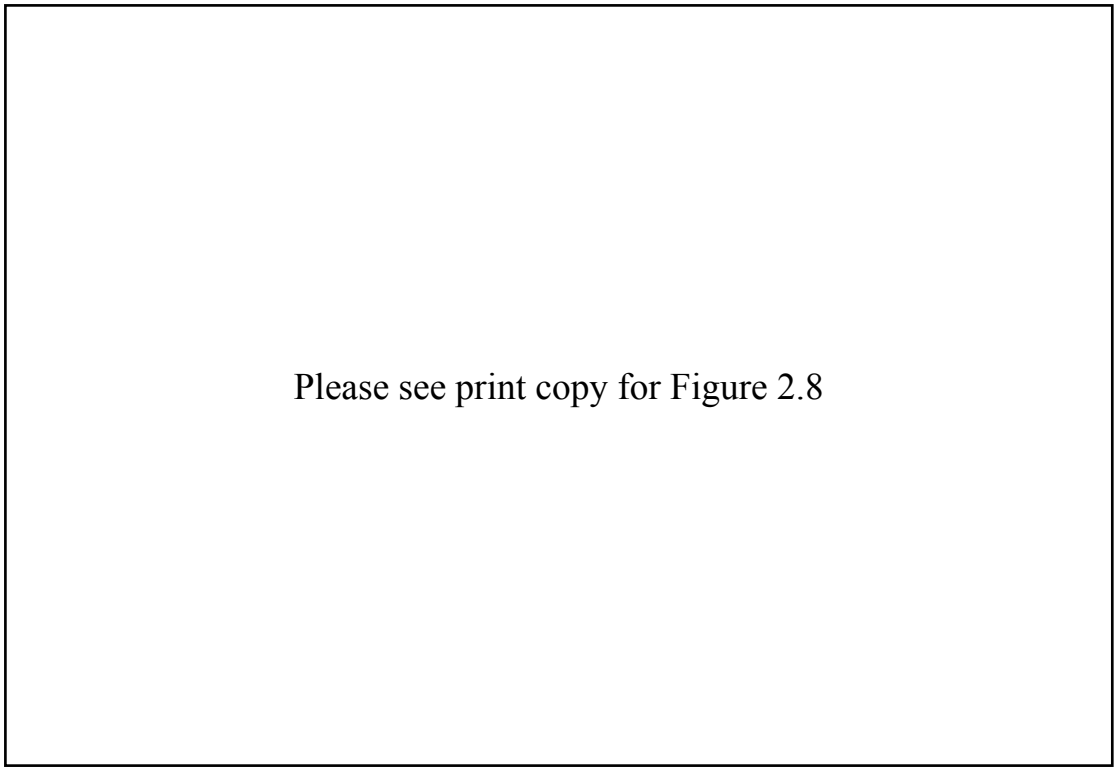


Figure 2.8: Wavelet decomposition of `lena.pgm`.

Vector Quantization

Vector quantization is used for image and sound compression. A vector quantizer divides samples into blocks and quantizes the blocks. There are various vector quantization methods. One of the methods is described as follows.

To quantize samples using vector quantization, first sample data (i.e. pixels in an image) is partitioned into blocks of k consecutive samples. Let X_i denote a k dimensional vector consisting of k consecutive samples. Then all X_i are in a k dimensional space. Let v_j denote k dimensional vector where $j \in \{1, 2, ..., n\}$. Then the k dimensional space is partitioned into n sub-spaces and v_j is assigned to each of the n sub-spaces. The vector v_j is chosen such that the Euclidean distance of v_j and X_i in the sub-space corresponding to v_j is minimized. A set of v_j is called *codebook*. Each X_i can be represented by j which is the index to a vector in the codebook. In de-quantization, to obtain k samples v_j is obtained from j using the same codebook in quantization.

In the following sections, the most commonly used standards are described.

2.5.3 JPEG

JPEG compression is one of the most commonly used image compression systems. The JPEG standard was prepared by CCITT study Group VIII and the Joint Photographic Experts Group (JPEG) of ISO/IEC JTC 1/SC 29/WG 10 [45]. The encoder of JPEG consists of three stages.

1. Discrete Cosine Transform
2. Quantization
3. Entropy coding

The above three stages are described in the following sections.

Discrete Cosine Transform

JPEG uses the 2-dimensional discrete cosine transform. An image is divided into 8×8 pixel blocks and an 8×8 DCT is applied to each of the blocks.

Quantization

The decomposed coefficients are quantized based on the *quantization table*.

Let h be an integer and r be a real number where $-0.5 \leq r < 0.5$, so that any real number can be shown as $h + r$. Let an integer rounding function be $rint$, that is,

$$rint(h + r) = h \ . \tag{2.14}$$

The quantization table is given by an 8×8 matrix where the entries are $Q^{(u,v)} \in \mathbb{N}$, $u, v \in \{1, 2, ..., 8\}$. Let $F^{(u,v)}$ and $T^{(u,v)}$ be the original and quantized coefficients, then $T^{(u,v)}$ is given by

$$T^{(u,v)} \quad = \quad rint(\frac{F^{(u,v)}}{Q^{(u,v)}}) \ . \tag{2.15}$$

The de-quantization of $T^{(u,v)}$ is given by,

$$\tilde{F}^{(u,v)} \quad = \quad T^{(u,v)} \cdot Q^{(u,v)} \ . \tag{2.16}$$

Entropy Coding

The quantized coefficients are entropy coded by either a Huffman or an arithmetic coder. For the DC coefficients, the differential DC values of two consecutive blocks are calculated and entropy-coded. For the AC coefficients, the 8×8 matrix of the quantized coefficients are scanned in a zig-zag order. In a scan, 63 coefficients are encoded by repeating the following procedure. The run-length of zero coefficients preceding a non-zero coefficient is obtained and then the pair of the run-length and the non-zero coefficient are entropy-coded. If the run of zero coefficients includes the last AC coefficient in the block, then the *end of block* code is encoded instead of the run-length and non-zero coefficient pair.

Compression Modes

JPEG provides lossy and lossless compression. With lossy compression, it provides different *modes* which determine the organization of encoded stream. We review the most commonly used two modes, *i*) *sequential DCT* and *ii*) *progressive* modes. In the sequential DCT mode, the organization of the encoded stream is such that each of 8×8 blocks consisting of 64 DCT coefficients are encoded sequentially. Assuming that an image consists of 8×8 pixel blocks, the order of scanning the blocks is from left to right in a row of 8×8 blocks, from the top to the bottom row.

The progressive mode provides two different encoding methods, *spectral selection* and *successive approximation*. In the spectral selection, all DC coefficients are encoded in the first *scan*. Then in the ith scan all the ith AC coefficients are encoded for $i = 1$ to 63. In each scan, the blocks are scanned from left to right in a row of 8×8 blocks, from the top to the bottom row.

Let L denote the precision of the quantized DCT coefficients. Then in the successive approximation, in the first *scan* all DC coefficients are encoded. Then from the most significant bits ($j = 1$) to the least ($j = L$), the jth bit layer of all AC coefficients are encoded.

2.5.4 JPEG2000

The JPEG 2000 encoding procedure can be decomposed into three stages [43], namely;

1. Transformation
2. Quantization

3. Embedded Block Coding with Optimized Truncation (*EBCOT*) [108, 109]

Discrete Wavelet Transform

In the transformation stage, an image is decomposed into *subbands* which are represented by real-valued wavelet coefficient sets.

Quantization

The transformation stage is followed by a *quantization* stage which converts the real-valued coefficients to whole numbers.

Embedded Block Coding with Optimized Truncation

Finally the coefficients are entropy-coded by *Embedded Block Coding with Optimized Truncation* to compress the output of the quantizer. In the EBCOT stage, each subband is divided into blocks and then each block is independently encoded into the bit-stream using an adaptive binary arithmetic coder in such a way that the more important information always precedes less important information. This is the heart of the embedded bit-stream organization.

2.5.5 MPEG

MPEG [42] video stream consists of three layers: video, audio and a system to interleave the two streams. The video and audio layers are independent from each other. We are only interested in the video layer. A video stream consists of a sequence of images, also called *frames*. Redundancy in the sequence is of two forms [30]: *i*) spatial redundancy which is due to redundancy in each frame, and *ii*) temporal redundancy that is due to similarities between consecutive frames. To compress the stream *transform coding* is used for the former while *motion compensation* technique is used for the latter. The final stage is an entropy coder that removes the remaining redundancy in the stream. In an MPEG stream, the image sequence is encoded as a sequence of *intra*, *forward predicted*, and *bidirectional prediction* frames. An *intra frame (I-frame)* is encoded without reference to any other frames; a *forward predicted frame (P-frame)* is encoded relative to the past reference I- or P- frame and a *bidirectional prediction frame (B-frame)* is encoded relative to the past and/or future reference I- or P- frames [115]. Frames are divided into 16×16 *macroblocks*, each consisting of 4 luminance and 2 chrominance 8×8 *blocks*.

In an I-frame, the transform coding is performed on the macroblocks. Each 8×8 block in a macroblock of an I-frame is transformed into 64 coefficients using Discrete Cosine Transform. This is followed by a scalar quantizer that replaces each DCT coefficient with an integer depending on the quantization scale. Finally the 64 quantized coefficients are *"zig-zag" scanned* to form a stream which is entropy coded.

In the P- and B- frame, to encode a macroblock, if an area similar to the macroblock is found in a past or a future frame, then a motion vector that refers to the similar area, and an error term between the macroblock and the area are encoded. This *motion compensation* reduces the temporal redundancy. If there is no such area in a past or a future frame, the macroblock is transform-coded in the same way as I-frame, i.e. without reference to any past or future frame.

2.6 Conclusion

For efficient transmission and storage, in most cases digital images are compressed either in a lossless or lossy way. Image compression systems compress digital images by removing the spatial redundancy and video compression systems also remove the temporal redundancy to obtain a stream with little redundancy. JPEG and MPEG are the two widely used international standards. In all these standards, digital images or frames are transformed and then quantization and entropy coding is used to obtain the compressed data. JPEG and MPEG use DCT, and JPEG2000 uses DWT. The more recent standard for image compression is JPEG2000. However both JPEG and JPEG2000 will be used for the foreseeable future.

Encrypting compressed streams will provide increased security. However, in general the size of the data is still large and using encryption algorithms such as RSA and DES is computationally expensive. This motivates development of combined encryption and compression systems that reduce the computational cost while providing reasonable security.

Lossy compression systems alter the bit representation of the original image but *the meaning of the image* will not be affected. This means that integrity checking algorithms must tolerate compression and decompression of image data. Cryptographic integrity checking algorithms are sensitive to a single bit change in the data and are not suitable for image authentication.

In the following chapters we propose combined encryption and compression, and authentication algorithms for digital images.

Chapter 3

Review of Image Encryption and Image Authentication Systems

3.1 Introduction

In this chapter we first review existing combined encryption and compression systems. If a data compression algorithm can be made to also provide security, less processing overhead could be expected as a single algorithm achieves two goals. First we review arithmetic coding encryption systems and next image encryption systems. Then we look at image authentication systems and finally we conclude.

3.2 Arithmetic Coding Encryption Systems

A method to integrate encryption and arithmetic coding was proposed by Witten and Cleary [120]. Adaptive arithmetic coding encryption schemes were motivated by the following observations.

1. *Models for data compression are often very large and may act as an enormous key.*

2. *If an adaptive model is used, the key depends on the entire transmitted text and finding the key would require tracking the changes to the model by decoding the entire transmission since initialization.*

3. *It is very difficult to regain synchronization if the models for compression and decompression are different.*

The proposed arithmetic coding encryption schemes are symmetric encryption algorithms. Depending on which part of the arithmetic encoder/decoder contributes to the key, the schemes are divided into two categories, which are *model-based schemes*

and *coder-based schemes*. Also there exists a scheme which combines the two. We describe these schemes in the following sections.

3.2.1 Model-based Schemes

In Witten *et al.*'s proposal [120], the model is used as the encryption key: that is the details of the model are only known to the transmitter and the receiver. These schemes are called *model-based*.

Model-based Scheme 1

The secret key of the scheme is *the initial model*.

In this scheme, the initial model is transmitted through a secure channel and is shared by the transmitter and the receiver. The other parameters such as the initial range of the coder is public. The secret are the parameters such as the initial frequencies of symbols and the order of symbols in the frequency table.

Model-based Scheme 2

The secret key of the scheme is *an initial string*.

The initial model and range are public and the key is a secret string, shared by the transmitter and receiver. The key is input to the system before the actual message is started. The initial string is sent through a secure channel preceding the transmission of a message. The key string modifies the models and the ranges in both the encoder and the decoder to one which is unknown to the attacker.

3.2.2 Coder-based Schemes

An alternative approach proposed by Irvine, Cleary and Rinsma-Melchert is a *coder-based* scheme [41].

The secret key of the scheme is *a bit string which is used to narrow the range*.

Based on the key bit sequence, either the high value h is decreased or the low value l is increased by an amount $(h-l)\varepsilon$ where ε is a public parameter and $0 < \varepsilon < 1$.

The known parameter ε can be a part of the key but it must be carefully chosen so as not to affect the compression performance. A variation of this scheme is proposed by Liu *et al.* [62].

3.2.3 Effect on Data Compression Performance

One important criteria for the performance of an arithmetic encoding encryption schemes is the effect of the added encryption on the efficiency of data compression systems. This is measured by the compression ratio, which is the average number of bits per input symbol, and the speed, which is the average time to process an input symbol. In both of the model-based schemes, the initial model or the initial string will be randomly chosen. Hence the given model for a message does not necessarily represent the statistics of symbols in the message. If an adaptive model is used, as data compression proceeds, the initial model is overwritten by the statistics of the incoming message. Witten *et al.* estimated that 1,000 symbols are enough for order-0 adaptive models to adjust the model to the input message. So if the length of a message is considerably longer than 1,000, the impact of a randomly chosen model or initial string on the data compression performance will be small.

In the case of coder-based schemes, the added encryption algorithm has a continuous influence on the data compression. The drop of the compression ratio may be minimized by an appropriate choice of ε. However, the scheme will have reduced compression speed because the extra-narrowing is equivalent to encoding another additional symbol. For example, the combined scheme by Liu *et al.* [62] results in an approximately 2% drop in the compression rate and almost doubles the coding time.

3.2.4 Security

All known attacks are chosen plaintext attacks. Attacks on adaptive schemes [11, 113] are chosen plaintext attacks where the attacker can feed plaintexts of her/his choice to the encoder. The attack does not discover the key (initial model) but succeeds to modify the model into a form which is known to the attacker, and hence allowing the attacker to decrypt the communication afterwords. By sending a long sequence of a single symbol to the encoder, the adaptive model is modified such that the probability of the symbol sent is maximized and that of other symbols are minimized and so the model is known to the attacker. Similar methods can be used to attack combined schemes [112].

To protect against this attack, a parameter of the model (e.g. re-calculation cycle of symbol probabilities) can be regularly changed, for example every n input symbols [11]. Another alternative is to use an additional simple operation such as a random transposition using a pseudo random generator [112].

3.3 Image Encryption

Advances in computer and communication technologies have resulted in efficient systems for delivery of a wide range of multimedia data such as video on demand and pay TV over the Internet. One of the main obstacles in the wide spread deployment of multimedia services has been enforcing security and ensuring authorized access to the data. A naive solution is to use an encryption algorithm to mask multimedia data streams. However direct application of encryption algorithms to multimedia data requires high computational power and introduces an unacceptable delay in communication. To alleviate these problems there have been numerous attempts to design encryption systems that take advantage of characteristics of this type of data and result in less expensive systems. One approach in this category has been incorporating encryption in the compression algorithm applied to the raw data. Because of the enormous size of multimedia data files and their high level of redundancy, in almost all cases the data is compressed before it is stored or transmitted and so incorporating security in the compression system is a very attractive approach. The main challenge is to ensure reasonable security without reducing the compression performance.

Image data in compressed format is still large and so using conventional encryption algorithm such as RSA and DES is computationally expensive. By combining image compression and encryption the computational cost can be reduced without compromising security. There are two approaches to achieve this : *i*) *elementary cryptographic operation* and *ii*) *selective encryption.*

3.3.1 Elementary Cryptographic Operations

Using less expensive elementary cryptographic operations such as *random permutation lists*, images can be effectively hidden. Since the operations are simple, the encryption will not have a high computational cost. An example of this approach is an MPEG (Motion Picture Experts Group) [42, 30] encryption system by L. Tang [107] that uses random permutation lists. In the following sections, we review two MPEG encryption systems, a system by L. Tang and its variant by S. U. Shin *et al.* [95].

MPEG Encryption System using Random Permutation Lists

In an MPEG system a motion picture consists of I-, P- and B-frames. I-frames are encoded independently from other types of frames. To encode an I-frame, the frame is divided into macro-blocks, each consisting of 8×8 pixel blocks. The pixel blocks

are transformed using an 8×8 DCT and then the resulting coefficients are quantized. The quantized coefficients in an 8×8 block are scanned in a zig-zag order as shown in Figure 3.1, and then are entropy-coded. In the P- and B-frame, the frame is divided into macro-blocks and each macro-block is encoded either using *motion compensation* or using the same method of encoding a macro-block in I-frames. In motion compensation, a pair of *an error term and a motion vector* that refers to a 16×16 area in a past (P- and B-frames) or a future (B-frame only) frame is encoded. Motion compensation is used if a region in a past or a future frame matches a macro-block in the P- and B-frame. Otherwise the macro-block is encoded in the same way as encoding the macro-blocks in I-frames. In this case, the macro-block is called *I-macro-block* [115].

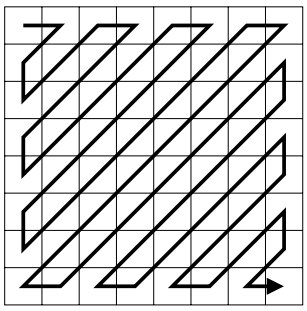


Figure 3.1: Zig-zag scan of 8×8 DCT coefficients in JPEG.

The encryption system proposed by L. Tang [107] encrypts macro-blocks in I-frames and I-macro-blocks in P- and B-frames. To encrypt an 8×8 DCT block after quantization, the zig-zag scan is replaced by a random scan using a *random permutation list* which is generated from a secret key and specifies the order of the scan such that the data is randomly scanned. The method of encrypting an 8×8 block is as follows.

1. First the following procedure is used for the DC coefficient. Denote the DC coefficient by an i digit binary number $b_i b_{i-1} ... b_2 b_1$. Then the value $b_i b_{i-1} ... b_{\frac{i}{2}+1}$ replaces the original value of the last (63rd) AC coefficient, i.e. the value of the AC coefficient will be lost, and the value $b_{\frac{i}{2}} ... b_2 b_1$ is set to the DC coefficient. This is called the *splitting procedure*. This operation is required because a DC coefficient in general is the largest among the 64 coefficients and cannot be hidden by the permutation.

2. Next all 64 DCT coefficients in the block are scanned using a random permutation list and then they are entropy-coded.

The author also suggested that the encryption method can be used in the JPEG

(Joint Photographic Experts Group) [45] compression system because JPEG compression system uses an 8×8 DCT and a zig-zag scan similar to MPEG.

MPEG Encryption System by S. U. Shin *et al.*

An MPEG encryption system [95] uses both random permutation lists and selective encryption. For all AC coefficients, the system replaces the zig-zag scan by the random scan using random permutation lists similar to Tang's system. For DC coefficients, the sign bits of the differences of two consecutive DC coefficients are encrypted using an encryption algorithm such as DES [38], RC4 [85, 86] and RC5 [96]. (The mode of DES is not specified in the paper but any mode can be used.)

3.3.2 Selective Encryption

It is possible to reduce the cost of computation by reducing the size of the data that is to be encrypted. That can be achieved by encrypting only part of the data or parameters that are required to decode images. For effective encryption, the part that is to be encrypted must be carefully chosen.

Aegis

An MPEG encryption system called *Aegis* (named after the breastplate of Zeus) [99] encrypts the MPEG video sequence header and I-frames in the MPEG stream using DES. The reasons for encrypting the header and I-frames are as follows.

1. The MPEG video sequence header contains the information required to initialize the decoder such as picture size (width and height), frame rate, bit rate and buffer size and the decoder needs this information to correctly decode the subsequent data.

2. I-frames are more important than P- and B-frames because P- and B-frames may have reference to I-frames but each I-frame is independent from other frames and so without I-frames, P- and B-frames will not be decoded.

SECMPEG

A video encryption system *SECMPEG* [69] uses DES (CBC-mode) to encrypt a video stream similar to MPEG-I. The video stream consists of the data of the MPEG-I stream and additional data which contains parameters for encryption and integrity check.

The organization of MPEG-I video stream consists of the following six layers [88].

Sequence layer This layer represents the highest level of a video sequence. A video sequence is defined as a sequence of a *sequence header*, one or more group of pictures (GOP) that follow the header, and a *sequence end code* at the end.

GOP layer This layer represents the organization of a group of pictures. A group of pictures consists of a *GOP header* and one or more frames.

Picture layer In this layer a frame consists of a *picture header* and one or more slices.

Slice layer A slice is a sequence of a *slice header* and one or more macro-blocks.

Macro-block layer A macro-block consists of a *macro-block header* and 8×8 blocks. In a typical color video, it includes four luminance blocks and two chrominance blocks.

Block layer This layer is only for a block in an I-macro-block and includes the entropy coded difference of consecutive two DC coefficients and run-length coded AC coefficients similar to JPEG.

The system provides four levels of security by choosing data in different layers for encryption. The four security levels are as follows.

Level 1 Picture headers, slice headers and GOP headers are encrypted.

Level 2 Addition to the parts encrypted in Level 1, macro-block headers and part of block data are encrypted.

Level 3 Addition to the parts encrypted in Level 2, all I-macro-blocks are encrypted.

Level 4 Entire MPEG stream is encrypted.

JPEG Encryption Systems by H. Cheng *et al.*

H. Cheng *et al.* proposed two systems [14]. One of the systems is based on the JPEG compression system. The system encrypts an image by dividing the 64 DCT coefficients in an 8×8 block into two parts, *i*) a lower frequency part and *ii*) a higher frequency part, and encrypts the lower frequency part.

The other system uses quad-tree image decomposition in the spatial domain [18, 105, 97] and is not based on a commonly used image compression system. The decomposed image (using quad-tree method) consists of two parts, *i*) a tree structure that

contains the location and size of rectangular regions, and *ii*) pixel values in the regions. The encryption system encrypts the pixel values because they are crucial to recover the image.

VEA by L. Qiao *et al.*

Video Encryption Algorithm (VEA) by L. Qiao *et al.* [79] is an MPEG encryption system and encrypts I-frames as follows.

1. Randomly permute 64 1's and 64 0's and generate a 128 bit number K.

2. Let $a_1a_2a_3...a_{128}$ denote a 128 byte sequence representing an entropy-coded I-frame. Then construct two 64 byte sequences, $\mathcal{M}_0$ and $\mathcal{M}_1$ as follows.

	Algorithm 1 : Construction of *Odd List* and *Even List*
1:	Initially $\mathcal{M}_0$ and $\mathcal{M}_1$ are zero bytes.
2:	For $i \in \{1, 2, ..., 128\}$
3:	If ith bit of K is 1
4:	a_i is concatenated to $\mathcal{M}_1$.
5:	Else
6:	a_i is concatenated to $\mathcal{M}_0$.

3. Then encryption of $a_1a_2a_3...a_{128}$ is given by $\mathcal{M}_0 \oplus \mathcal{M}_1 + E_{DES}(\mathcal{M}_0)$, where $X \oplus Y$ is XOR of X and Y, $X+Y$ is concatenation of X and Y and $E_{DES}(X)$ is encryption of X using DES.

VEA by C. Shi *et al.*

Video Encryption Algorithm (VEA) by C. Shi *et al.* [92] is an MPEG encryption system. It encrypts sign bits of DCT coefficients (for the DC coefficients, the sign bits of differences of DC coefficients) in I-macro-blocks using the XOR operation of the sign bits and a random bit string.

RVEA by C. Shi *et al.*

Real-time Video Encryption Algorithm (RVEA) by C. Shi *et al.* [94] is an MPEG encryption system and encrypts sign bits of motion vectors in addition to the encryption of sign bits of DCT coefficients in VEA [92] using DES or IDEA [53].

3.3.3 Compression Performance of Encryption Systems

Experimental results of Tang's system are shown in his paper[107]. For two MPEG encoded streams "flower garden" and "table tennis", the increase in encoding time for encryption was -0.2% and 0.6%, and the increase in the size of the encoded stream was 21.9% and 41.9%, respectively. Hence, it can be seen that encryption using random permutation lists is fast and the drop in the encoding speed is ignorable. However the drop in compression rate is very large.

The results of experiments [95] show that the system proposed by S. U. Shin *et al.* increased the encoded stream size by 0.3% on average. The results show that 1.2% of the entire MPEG stream was encrypted. We note that although the system uses the same method as Tang's method [107] to encrypt AC coefficients, the drop in compression rate of their system [95] and Tang's system [107] has a large difference.

The large drop in compression rate is due to the permutation of the AC coefficients. Without this permutation, the high frequency AC coefficients are scanned consecutively in the zig-zag order. Most of these coefficients are likely to be zero and so are run-length coded. If the coefficients are encrypted, the random permutation will result in a shorter run-length. The contribution of the run-length coding to the compression rate is large [88] and so is the drop in the compression rate.

The results of experiments for SECMPEG [69] show that the increase in computation time for encryption is about 10%, 12% and 55% for security levels 1, 2 and 3, respectively. In security level 3, SECMPEG encrypts picture headers, slice headers, GOP headers, macro-block headers and I-macro-blocks. VEA by L. Qiao *et al.* encrypts I-frames, which is about 50% of the entire MPEG stream [79], using DES. The comparison of VEA with encryption of the entire stream using IDEA shows that the encryption speed of VEA is approximately 50% of IDEA [79]. VEA by C. Shi *et al.* increases the encoding time by 1.81% for encrypting sign bits of DCT coefficients in I-frames using DES. RVEA encrypts sign bits of motion vectors and DCT coefficients using DES. About 10% of the MPEG stream is encrypted and the encryption increases the encoding time by 2.55%.

The data size will largely affect the encoding and decoding speed. Encryption data size largely varies with an encryption system. For example, to encrypt an I-frame, VEA by L. Qiao *et al.* encrypts the entire I-frame data (50% of the entire stream) but RVEA encrypts only sign bits of the motion vectors together with the DCT coefficients (10% of the entire stream).

3.3.4 Security

Encryption Using Random Permutation Lists

Analysis by L. Qiao *et al.* [81, 80] showed that encryption using random permutation lists are vulnerable to a known-plaintext attack. Since many movies start with standard header clips, such as the MGM roaring lion, an attacker can compare the original DCT coefficients with the permuted ones to find the permutation. The analysis also pointed out that the ciphertext only attack can reveal the images. It showed that the DC coefficient is the largest among all DCT coefficients in a block and non-zero AC coefficients are gathered in the low frequency part. Hence the correct order of coefficients in a block can be recovered by sorting them.

The images in Figure 3.2 and 3.3 show the reconstructed images by sorting the quantized DCT coefficients. Each image was DCT transformed and uniform-quantized. In the images in Figure 3.2, the 16 largest coefficients were assigned to the 16 lowest frequencies such that the larger coefficient was assigned to the lower frequency. The other 48 frequencies were set to zero. In the images in Figure 3.3 the sorted 64 coefficients were used where the largest coefficient was assigned to the DC. Then the images were reconstructed by de-quantizing and inverse-transforming these coefficients. The PNSRs of the reconstructed images are given in Table 3.1. The experiments clearly show that images can be recovered although the image quality can be low.

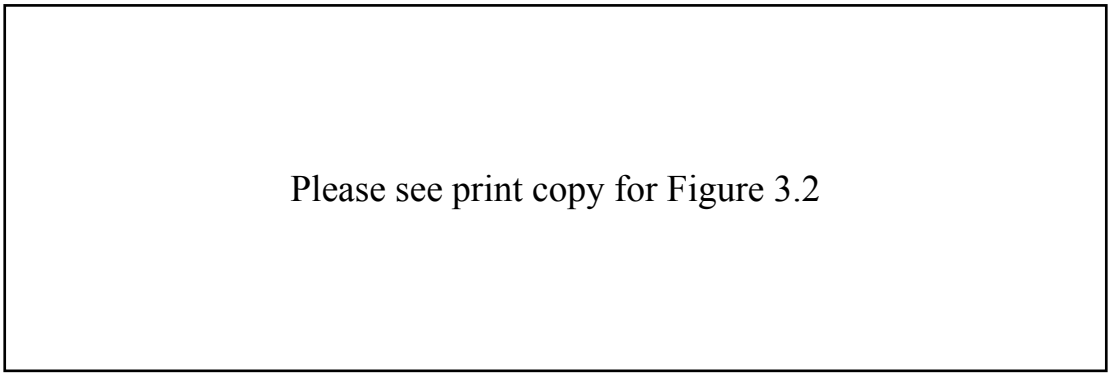
Table 3.1: PSNR of reconstructed images `lena`, `mandoril` and `peppers` by sorting DCT coefficients : using largest 16 coefficients and 64 coefficients.

	`lena`	`mandoril`	`peppers`
largest 16 coefficients	23 dB	17 dB	18 dB
64 coefficients	13 dB	11 dB	11 dB

Selective Encryption

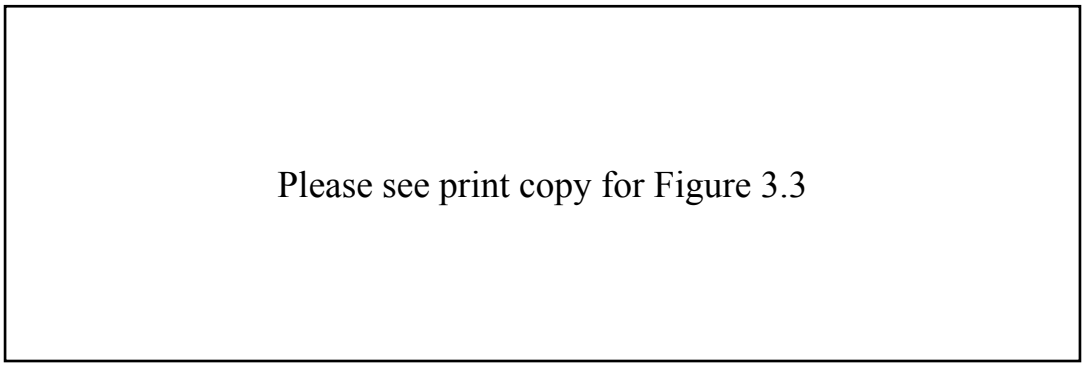
A JPEG encryption system by H. Cheng *et al.* [14] only encrypts the lower frequency coefficients. The authors had concluded that the system is insecure because the edges of the encrypted image are contained in the higher frequency part and so will not be hidden.

In [2] I. Agi *et al.* showed that MPEG encryption systems which encrypt only I-frames are insecure. P- and B-frames may include I-macro-blocks which are independently decodable and the I-macro-blocks are not encrypted and so they will reveal the images. The authors had pointed out that the number of I-macro-blocks in P- and



(a) (b) (c)

Figure 3.2: Reconstructed images using sorted largest 16 coefficients : `lena` (a), `mandoril` (b) and `peppers` (c).



(a) (b) (c)

Figure 3.3: Reconstructed images using sorted 64 coefficients : `lena` (a), `mandoril` (b) and `peppers` (c).

B-frames can be large if the scene includes high degree of motion. Another experiment using "Miss America II", where Miss America is sitting behind a desk and speaking to camera, shows that even if all I-macro-blocks are encrypted, the stream can reveal some features of the person. The authors of [2] concluded that encryption systems that encrypt I-macro-blocks but do not encrypt motion vectors, such as Aegis and SECM-PEG, are not suitable for applications which require a high level of security. We point out that the two VEA by L. Qiao *et al.* and by C. Shi *et al.* do not encrypt motion vectors and so will not provide a high level of security.

3.3.5 Concluding Remarks

It can be seen from above that encryption using simple operations has little impact on the encoding speed. However, using random permutation lists with MPEG and JPEG

does not provide high security because the lower frequencies contain higher energy and so larger coefficient values. Obviously it is inappropriate to apply permutation to such data since the coefficient values and their frequencies have strong correlation. We note that the MPEG and JPEG compression algorithms exploit the correlation to compress the data. Permutation destroys the correlation and results in a large drop in the compression rate.

Selective encryption usually has a high computation cost and the drop in the compression speed is determined by the size of the encrypted data. The advantage of selective encryption is that well-studied encryption algorithms can be used for encryption and so if the parts to be encrypted are carefully chosen, high security can obtained. If selective encryption is applied to entropy-coded data (e.g. SECMPEG and VEA by L. Qiao *et al.*), there will be no drop in compression rate.

It can be seen from the above that to assess the security of image encryption systems it is important to understand properties of MPEG and JPEG coded data. From the security point of view, more analysis of the systems to assess the level of security against various attacks is required.

3.4 Image Authentication

In recent years there has been a rapid increase in on-line multimedia services. Visual data and in particular images have become part of nearly all Web pages. In many applications, data must be authenticated. For example, images used in news reporting or taken by a speed camera must be authenticated.

Authentication of image data poses many new challenges. Firstly, unlike data authentication systems that must detect a single bit change in the data, image authentication systems must remain tolerant to a range of modifications that are due to commonly used operations on such data, including filtering operations for enhancement and lossy compression to reduce the size of the data by removing irrelevant information (that is, details which are not perceptible) from it. In all, the above resulting object will have different pixel values from the original, but it remains perceptually the same. Objects may be decompressed and re-compressed with a different quality level and still must remain verifiable if the object is not tampered with. In other words an authentication system must be able to distinguish between acceptable and not acceptable changes and allow the verification to succeed or fail, depending on the two cases, respectively. Secondly, because of the very large size of data files, and in many

cases real-time nature of the data, very efficient systems are required. That is, the authentication algorithm should only add a small overhead to the multimedia delivery system.

An image authentication system consists of two algorithms, *i*) an authentication algorithm which takes an image and some key information and generates an authenticated image, and *ii*) a verification algorithm that takes a candidate image and the key information, and produces a true or a false result. The verification algorithm must not require the original image to verify a candidate image and should be able to localize the modified part if a candidate image is tampered. If the original image is available in the verification process, then a given image can be verified by comparing to the original and so watermarks or signatures are not required. However, the original image must be securely transmitted to the verification system and this is costly compared to transmitting signatures or keys of watermarks which are significantly smaller than the image.

In some systems the authentication and the verification algorithms do not share secret information and all data which the verification algorithm requires are public. In other systems secret information must be transmitted to the verification system using a secure channel.

Image authentication systems can be broadly divided into *watermarking system* and *signature system*. In the following we review these systems.

3.4.1 Watermarking Systems

In a watermarking system *a watermark* [124] signal is embedded into the image such that it can be recovered even if the image undergoes a set of predefined operations [128, 67]. Watermarking schemes are divided into two classes, *i*) robust watermarks, which resist various image manipulations and *ii*) fragile watermarks, which are sensitive to change in an image. For image authentication, watermarks must detect changes and so fragile watermarks are used.

A watermark signal can be embedded in the pixel domain or in the transform domain using an algorithm such as the Discrete Cosine Transform (DCT) or the Discrete Wavelet Transform (DWT). Watermarks would be required to survive image compression such as JPEG because it is most likely that images are compressed for efficient transmission and storage. If the watermarking algorithm uses the same transformation as an image compression system, it can be integrated into the compression system so

that the image in the compressed form carries the watermark. Otherwise the watermark should survive image compression in general.

In the following text, we review a number of recently proposed fragile watermarking systems to give examples of this technique. A more comprehensive information of image watermarking can be found in the papers [128, 67, 106, 98].

Watermarking System by Wu *et al.*

Wu and Liu [124] proposed an authentication system in which a watermark is embedded in the image by changing the DCT coefficients after the quantization phase of JPEG compression. In the system, all possible DCT coefficient values are flagged as either 0 or 1, and the value of a coefficient corresponding to the flag is used to embed a bit. If the embedding bit matches the flag of the corresponding DCT coefficient value, the coefficient is not modified and if it does not, the coefficient is modified to the closest value, the flag of which matches the embedding bit. Embedding a watermark into the DC and the low energy AC coefficients would result in blocking artifacts. The system can locate the modified regions of still images and motion pictures.

Watermarking System by P. W. Wong

In the *Public Key Watermark* system [122], an embedding signal is generated by calculating exclusive-or of a bi-level watermark image and the hash value using MD5 [84] from the original image. The signal is digitally signed by the private key of a public key cryptosystem and is embedded in the least significant bits of pixels in the original image. The verification system extracts the embedded signal, and verifies it using the public key. Then it calculates the hash value from the candidate image and recovers the bi-level watermark image by calculating exclusive-or of the embedded signal and the hash value. In this system the authentication algorithm uses only public information and does not share secret information with the authentication algorithm. Since the system uses MD5 and a public key cryptosystem that is sensitive to single bit changes, the watermark will not survive lossy compression.

Watermarking System by J. Fridrich

The watermarking system [29] embeds watermark into the DCT coefficients. Assuming that we embed r symbols of length m, the algorithm for embedding the watermark is as follows.

1. Divide the original image into 64×64 blocks B_i where $i \in \{1, 2, ..., n\}$.

2. Generate 64×64 black-and-white patterns P_j where $j \in \{1, 2, ..., m\}$ by a pseudorandom number generator (PRNG) using a secret key K.

3. Then smooth P_j using a low-pass filter and make them DC-free.

4. Let t denote a threshold value. Then obtain m bits from B_i using P_j using the following algorithm.

	Algorithm 2 : Obtain bits from blocks
1:	For each block B_i,$i \in \{1, 2, ..., n\}$
2:	For each pattern P_j,$j \in \{1, 2, ..., m\}$
3:	If $\lvert P_j \cdot B_i \rvert > t$
4:	$b_{ij} = 1$.
5:	Else
6:	$b_{ij} = 0$.

   The value of t is chosen so that approximately half of b_{ij} are ones. In Fridrich's paper [29], $t = 2500$ was used.

5. Generating and embedding watermark signal is as follows.

	Algorithm 3 : Generating and embedding watermark signal
1:	For each block B_i,$i \in \{1, 2, ..., n\}$
2:	Initially watermark signal S_i is zero.
3:	For m symbols
4:	Generate a pseudorandom sequences of length $D + r$ using PRNG with key K,i,j,and b_{ij}.
5:	According to a symbol to be embedded, choose a segment of length D from the sequence of length $D + r$ where there are r possible choices.
6:	Add the sequences of length D to S_i.
7:	Transform B_i using DCT.
8:	Choose the middle D DCT coefficients and add S_i to them.

The verification algorithm executes steps 1 to 4 above. To extract the embedded symbols in B_i, it generates a pseudorandom number sequence of $D + r$ similar to the watermark embedding in Algorithm 3. It calculates the cross-correlation of the shifted versions of the pseudorandom number sequence of length $D + r$ and D coefficients in

B_i. The embedded symbol is determined by the amount of shift which gives the largest correlation.

3.4.2 Signature Systems

The aim of a signature system is to extract some features (also called a signature, image digest, or *Message Authentication Code* (MAC)) of the image that remain invariant for images that have undergone predefined operations (e.g. JPEG compression to a given quality level). The signature will be appended to the image and so an authenticated image is a pair consisting of the image and a signature. For the generation and the verification of a signature, some systems use secret information. In these systems, the extracted features form the *MAC* or *authentication tag*.

In the following, we review a number of signature systems that use the data in different domains to generate signatures, e.g. pixels [89], DCT coefficients [58, 126], and wavelet coefficients [12].

Signature System by M. Schneider *et al.*

Schneider *et al.* [89] proposed a digital signature system for image authentication. In their paper [89], the authenticity measure using features of images is defined as follows. Let I_o and I_m be the original and a candidate image, and $g(I)$ be a function that computes a *feature vector* for image I. Then *feature authenticity* of I_o and I_m is given by

$$A_{feature} = 1 - ||g(I_o) - g(I_m)||$$

where $||X - Y||$ is the normalized distance between two feature vectors X and Y (the distance between X and Y divided by the maximum possible distance of two feature vectors). For example, $A_{feature}$ is 0 if $g(I_o)$ and $g(I_m)$ has the maximum distance, and $A_{feature}$ is 1.0 if $g(I_o) = g(I_m)$.

The system uses the intensity histogram to calculate the feature vector and public key encryption algorithm to sign the feature data. The generation of a signature is as follows.

Algorithm 4 : Generation of a signature	
1:	Compute a feature vector from I_o by $g(I_o)$.
2:	Compute $H(g(I_o))$ where $H()$ is a hash function.
3:	Sign $H(g(I_o))$ using the private key.

Then choose a threshold value τ which determines the degree of modification that is acceptable. The verification of a signature is as follows.

Algorithm 5 : Verification of a signature	
1:	Compute a feature vector from I_m by $g(I_m)$.
2:	Compute $H(g(I_m))$.
3:	Decrypt $H(g(I_o))$ using the public key.
4:	Compare $H(g(I_o))$ and $H(g(I_m))$.
5:	If the difference between $H(g(I_o))$ and $H(g(I_m))$ is equal to or smaller than τ
6:	output true (I_m is authentic).
7:	Else
8:	output false (I_m is not authentic).

The authors noted that if $\tau \neq 0$, then cryptographic hash functions cannot be used for $H()$ because they are sensitive to a single bit change.

To calculate the feature vector, the following method was used.

1. An image was divided into blocks.
2. The intensity histogram [31] was calculated for each block.
3. To calculate the distance of two feature vectors, Euclidean distance between intensity histograms was used.

The authors suggested to embed the signature into an image using watermarking techniques in the papers [65, 20].

SARI System by Lin *et al.*

C. Lin and S. Chang [58] proposed an authentication system in which an authenticated image remains authenticated after JPEG compression and decompression. The system exploits the fact that the relationship between DCT coefficients of the same frequency in two blocks is invariant over JPEG compression.

The system is based on the following theorem.

Theorem 1 *Assume* ${F_p}^{(u,v)}$ *and* ${F_q}^{(u,v)}$ *are DCT coefficients of two arbitrarily* 8×8 *non-overlapping blocks of image* X*, and* $Q^{(u,v)}$ *is the quantization table of JPEG lossy compression* $\forall u, v \in \{0, ..., 7\}$ *and* $p, q \in \{1, ..., \wp\}$*, where* $\wp$ *is the number of blocks in* X*. Define* $\Delta {F_{p,q}}^{(u,v)} = {F_p}^{(u,v)} - {F_q}^{(u,v)}$ *and* $\Delta \tilde{F}_{p,q}^{(u,v)} = \tilde{F}_p^{(u,v)} - \tilde{F}_q^{(u,v)}$ *where* $\tilde{F}_p^{(u,v)}$

is defined as $\tilde{F}_p^{(u,v)} = rint({F_p}^{(u,v)}/Q^{(u,v)})Q^{(u,v)}$ *where* $rint()$ *is an integer rounding function.*

Assume a fixed threshold $k \in \Re, \forall u, v$, *and define* $\tilde{k}^{(u,v)} = rint(\frac{k}{Q^{(u,v)}})$. *Then, if* ${\Delta F_{p,q}}^{(u,v)} > k$,

$$\Delta \tilde{F}_{p,q}^{(u,v)} uv \geq \begin{cases} \tilde{k}^{(u,v)} \cdot Q^{(u,v)}, & \frac{k}{Q^{(u,v)}} \in \mathbb{Z} \\ (\tilde{k}^{(u,v)} - 1) \cdot Q^{(u,v)}, & \frac{k}{Q^{(u,v)}} \notin \mathbb{Z} \end{cases}$$

else if ${\Delta F_{p,q}}^{(u,v)} < k$,

$$\Delta \tilde{F}_{p,q}^{(u,v)} uv \leq \begin{cases} \tilde{k}^{(u,v)} \cdot Q^{(u,v)}, & \frac{k}{Q^{(u,v)}} \in \mathbb{Z} \\ (\tilde{k}^{(u,v)} + 1) \cdot Q^{(u,v)}, & \frac{k}{Q^{(u,v)}} \notin \mathbb{Z} \end{cases}$$

else ${\Delta F_{p,q}}^{(u,v)} = k$,

$$\Delta \tilde{F}_{p,q}^{(u,v)} uv = \begin{cases} \tilde{k}^{(u,v)} \cdot Q^{(u,v)}, & \frac{k}{Q^{(u,v)}} \in \mathbb{Z} \\ \tilde{k}^{(u,v)} \cdot Q^{(u,v)} \ or & \\ (\tilde{k}^{(u,v)} \pm 1) \cdot Q^{(u,v)}, & \frac{k}{Q^{(u,v)}} \notin \mathbb{Z} \end{cases}.$$

■

The MAC is obtained by choosing a sequence of threshold values for each pair (u, v) and outputting one bit for each threshold value. The detail of the system is described in Chapter 7.

Feature Extraction by Bhattacherjee *et al.*

Bhattacherjee and Kutter [12] proposed an authentication system that uses feature extraction. The system transforms an image into wavelet coefficients using the Mexican-Hat wavelets [12].

Let $M_i(\vec{x})$ be a wavelet coefficient at position $\vec{x}$ in subband i. Then the feature detection function is given by

$$P_{ij}(\vec{x}) = |M_i(\vec{x}) - \gamma M_j(\vec{x})|$$

where $\gamma = 2^{-(i-j)}$. Let $N_{\vec{x}}$ denote a set of positions that are within radius r pixels of $\vec{x}$ where $r = 5$ was used [12]. Then procedures to calculate feature points are as follows.

1. Position $\vec{x}$ is a candidate of a *feature point* if $P_{ij}(\vec{x}) = \max_{\vec{x}' \in N_{\vec{x}}} P_{ij}(\vec{x}')$.

2. If the variance of the pixels in the $n \times n$ neighborhood of $\vec{x}$ is larger than a user-defined threshold, $\vec{x}$ is a feature point. In [12], $n = 7$ and 10 was used for the threshold.

The authentication system inputs the original image I_o and generates a set of feature points $S_o = \{\vec{x}_1, \vec{x}_2, ...\}$. The verification system inputs a candidate image I_m and S_o. It generates a set of feature points S_m from I_m and then compares $\vec{x} \in S_m$ with $\vec{y} \in S_o$. Two feature points are considered matched if $|\vec{x} - \vec{y}| < 2$. If all feature points in S_o and in S_m match then I_m is considered as authentic.

The experiments [12] showed that the system detected modified locations and survived JPEG compression of quality level 80%.

MAC by L. Xie *et al.*

Approximate Image Message Authentication Codes (IMACs) [126] uses *Approximate Message Authentication Code (AMAC)* [32, 6]. The AMAC algorithm is a probabilistic checksum which estimates the similarity of two messages using their Hamming distance [126]. Let K denote a secret key and m be an input binary sequence of length $l \times r \times s$ where l is the AMAC length and $r, s \in \mathbb{N}$. The AMAC length must be large to provide security and its typical value is $80 \leq l \leq 400$. The generation of AMAC can be divided into four stages, *i*) *initialization*, *ii*) *formatting*, *iii*) *randomization*, and *iv*) *majority bits calculations*.

Initialization Initialize a pseudorandom number generator (PRNG) using K.

Formatting From m construct a binary matrix M consisting of l columns and $r \times s$ rows.

Randomization The randomization algorithm is as follows.

	Algorithm 6 : Randomization
1:	Generate a random permutation list to permute l items using the PRNG.
2:	For all rows in M
3:	Permute each row using the random permutation list.
4:	Generate a $(r \cdot s) \times l$ binary matrix R consisting of random bits using the PRNG.
5:	Calculate $T = M \oplus R$ where $\oplus$ is the XOR operation.

Majority bits calculations A *majority bit* is defined as the most frequent binary symbol in a set of bits. This stage consists of two rounds of majority bits calculations.

The first round

In the first round, $l \times r \times s$ bits in T are reduced to $l \times s$ bits using majority bits calculation. Let V denote a $s \times l$ binary matrix. Then the following algorithm generates V.

Algorithm 7 : Majority bits calculation 1	
1:	Divide T into $l \times r$ matrices U_i,$i \in \{1, 2, ..., s\}$.
2:	For each U_i,$i \in \{1, 2, ..., s\}$
3:	For jth column of U_i,$j \in \{1, 2, ..., l\}$
4:	Obtain a majority bit from r bits in the column.
5:	Set the majority bit to position (j, i) in V.

The second round

In the second round, $l \times s$ bits in V are reduced to l bits. The algorithm is as follows.

Algorithm 8 : Majority bits calculation 2	
1:	For jth column of V,$j \in \{1, 2, ..., l\}$
2:	Obtain a majority bit from s bits in the column.
3:	Output the majority bit.

Let I_o be the original image. Then the authentication system generates the AMAC and the modified image I'_o, which is distributed instead of I_o. The algorithm is as follows.

1. Divide an image I into 8×8 blocks and obtain the DC coefficients using the DCT.

2. Construct a bit sequence from the most significant bits of the DC coefficients and calculate AMAC from the bit sequence using the secret key K.

3. Let t denote a user defined error tolerance value. Assuming that the DC coefficient is $[0, 255]$, divide the original DC coefficients in $[0, 255]$ into two sets $\mathcal{D}_l$ and $\mathcal{D}_h$ such that $0 \leq d_l \leq 127$ for $d_l \in \mathcal{D}_l$, and $128 \leq d_h \leq 255$ for $d_h \in \mathcal{D}_h$. Then d_l and d_h are modified as follows.

$$d'_l = \frac{127 - t}{127} d_l$$

$$d'_h = \frac{127 - t}{127}(d_h - 128) + 128 + t \ .$$

The modified coefficients d'_l and d'_h are within the range $[0, 127 - t]$ and $[128 + t, 255]$, respectively. The modification is used to prevent the MSBs of the DC

coefficients from changing due to acceptable operations such as JPEG compression. For example, if JPEG compression can change the values of coefficients by 1, then 127 may change to 128 and so the MSB changes from 0 to 1. If $t = 1$, all coefficients are either in $[0, 126]$ or $[129, 255]$ and the MSBs of coefficients will not change. In the JPEG compression case, the value t can be chosen to be $t = 0.5q$ where q is the JPEG quantization step for the DC coefficients.

4. Inverse-transform the modified DCT coefficients and construct a new image I'_o.

The verification system inputs a candidate image I_m, the AMAC, and secret key K. It calculates the AMAC from I_m similar to the authentication algorithm and compares it with the received AMAC. Using the Hamming distance of two AMACs, the authenticity of I_m is determined. The verification system can tolerate bit changes in the received image.

3.4.3 Evaluation

E. T. Lin *et al.* [60] give a list of desired properties of fragile watermarking systems. Some applications may only require some of them. The properties are as follows.

1. A system must detect any tampering with high probability.

2. An embedded watermark should not be visible by human eyes. This property is called *Perceptual Transparency*.

   To make the watermark resistant against acceptable operations such as compression, the commonly used method is to increase the level of embedding signals. This can degrade the image quality.

   Signature systems do not have this problem because they do not modify images (except [126]).

3. Detection should not require the original image. In many applications the original image may not be available because they are immediately watermarked when they are created. Since the original image will have a large size, storing and transmitting such data using a secure channel is inefficient.

4. Verification systems should be able to locate modifications.

5. The verification systems should be able to characterize modifications. It should be able to estimate the type of modifications such as the addition of edges. It seems that not many system have this ability.

6. Watermarks generated by different keys should be *orthogonal*. That is, an embedded watermark in an image that was generated by a particular key must be detected only by using that key.

7. The watermarking key space should be large.

8. The watermarking key should be difficult to deduce from public information.

9. The insertion of a watermark by unauthorized parties should be difficult.

10. The watermark should be embedded in the compressed domain. Conversely, the watermark should survive compression because images are commonly stored in compressed form.

In many papers the computational costs of the systems are not taken into account. For example, the wavelet transform is used in several systems [12, 125, 116] which makes the systems computationally more expensive than an 8×8 DCT.

M. Wu *et al.* [123] proposed an attack against *trusted devices* such as a scanner and a digital camera, which insert a new fragile watermark in the image. The attacker obtains an image and modifies it. Then he either scans the modified image using the trusted scanner which inserts a new watermark into the scanned image, or takes a picture of the modified image using a trusted digital camera which embeds a new watermark into the image taken. The authors suggested to use a pair of robust and fragile watermarks. The robust watermark will survive under the operations such as scanning or taking picture although the fragile one may not. The modified and then copied image will contain two watermarks, i.e. the original robust watermark and a new one inserted by the device and so it is possible to detect the modified one.

For signature systems, properties 1,3,4,5 in the above list are also appropriate. In addition to these properties, it should be difficult to find more than one distinct input images which will generate the same signature. That is, it should be difficult to find a collision for signatures. We note that the difficulty of finding the input image from the signature is not required for systems such as [12] although it is required for the systems such as SARI which allows attackers to find collisions once the relationship between the signature bits and pixels are revealed.

3.4.4 Concluding Remarks

The advantage of watermarking systems is that, unlike signature systems, there is no need for a separate authenticator as the image carries the authenticating information

with itself.

Watermarking systems that are used for authentication must be fragile. That is the watermark must be destroyed (become irrecoverable) with the slightest change to the image. However compression tolerance means that the watermark must survive changes that are due to JPEG compression algorithm. Reconciling these two requirements, that is fragility and compression tolerance, is a challenge that must be addressed in this context. A number of systems have been proposed but many of the ones that are based on fragile watermarks are less tolerant to JPEG compression [122, 29]. To make the watermark resistant against compression, the level of noise embedded into an image needs to be increased and so the image quality will be degraded.

Some watermarking systems such as [127] have been analyzed and shown to be insecure [75] but many systems remain with no real security modeling or analysis.

3.5 Conclusion

We reviewed various secure compression systems and image authentication systems. Many systems have been proposed but analysis of these systems has usually been ad hoc. To correctly assess security of these systems, further research is required.

In the following chapters we examine attacks and propose new secure image compression and image authentication systems.

Chapter 4

Attacks on Image Encryption Systems

4.1 Introduction

In recent years, numerous systems that incorporate encryption in the *MPEG* (*Moving Picture Experts Group*) [42] compression system have been proposed [55, 107, 79, 92, 93, 95, 94]. MPEG is one of the most widely used compression standards for video data. The proposed schemes use a range of approaches including selective encryption of parts of the stream and permutation of transform coefficients which can completely mask the data. The schemes can effectively reduce computation and delay but degrade the compression performance and offer differing degrees of security. Although it is straight forward to measure the drop in compression as a result of adding encryption, it is much harder to assess security of the systems. In particular it is misleading to use the length of the key as a measure of security.

In this chapter we show new attacks on the MPEG encryption systems. Firstly, we demonstrate an attack on MPEG encryption systems that use random permutation lists. Then we show a method to recover encrypted DC coefficients from AC coefficients and demonstrate that if AC coefficients are known, encrypting DC coefficients is ineffective. Finally, we conclude.

4.2 Chosen DCT Coefficients Attack on MPEG Encryption Schemes

In this section we present a chosen DCT coefficients attack against encryption systems by L. Tang [107] and S. U. Shin *et al.* [95], which use a random permutation of DCT coefficients. This approach is usually complemented by other methods to enhance the security. We show that by using a number of well chosen sequences of transform coefficients it is possible to derive the secret permutation.

Firstly, we briefly review the MPEG encoding and then outline encryption schemes using random permutation lists in Section 4.2.1. In Section 4.2.2 we present our attack and in Section 4.2.3 conclude the section.

4.2.1 Encryption Using Random Permutation

The basic approach in random permutation schemes used by Tang [107] and Shin *et al.*[95], is to replace the *zig-zag* scan with a random permutation. The key is used to select the permutation from a set of possible permutations and use it to read elements of an 8×8 array matrix of the quantized DCT coefficients of the 8×8 pixel block. This method is applied to all I-blocks [107] and only I-frames [95].

Since quantized DC coefficients are significantly larger than other coefficients, Tang [107] proposed splitting each DC coefficient into two 4-bit numbers corresponding to the most significant 4 bits and the least significant 4 bits respectively. One number is stored as the DC value and the other replaces the coefficient corresponding to the highest frequency. The highest frequency coefficient can be replaced because it is known that the visual importance of this coefficient is negligible. As additional security measures, [107] suggests encrypting DC coefficients using a block cipher such as DES by forming a block of eight DC coefficients. We note that DES is no longer considered to be secure [27, 37]. An alternative proposed method [95] is to encrypt the sign-bit of every DC coefficient in an 8×8 block. We will be mainly interested in random permutation.

Decryption

To decrypt an MPEG stream that is encrypted using random permutation, the inverse permutation is used to recover the original order of the 64 DCT coefficients and then the Inverse Discrete Cosine Transform (IDCT) is applied to the result.

Let $S = (s_i)$, $i \in \{0, 1, \ldots 63\}$ denote a 1×64 vector of input DCT coefficients, and $Q = (q_{i,j})$, $i, j \in \{0, 1, \ldots 63\}$ denote the 64×64 inverse permutation. Q is a zero-one matrix with exactly one non-zero entry in each row and column.

Let $B = (b_i), i \in \{0, 1, \ldots 63\}$, denote a 1×64 vector corresponding to the 64 inverse-permuted DCT coefficients. Then we have,

$$B = S \cdot Q \ . \tag{4.1}$$

For the variant [95] that does not permute the DC coefficient, the first row and column of Q are of the form : $q_{0,0} = 1$, $q_{0,j} = 0, j \in \{1, 2, \ldots 63\}$ and $q_{i,0} = 0, i \in$

$\{1, 2, \ldots 63\}$ and the sub-matrix $Q' = (q_{i,j})$, $i, j \in \{1, 2, \ldots 63\}$, is a permutation matrix of size 63.

Inverse DCT

The decoder performs an IDCT on the recovered DCT coefficients.

Let $\hat{C}$ denote the 8×8 transform matrix of the IDCT and $\hat{B}$ denote the 8×8 matrix defined from B where $\hat{b}_{i,j} = b_{8i+j}$, $i, j \in \{0, 1, \ldots 7\}$.

The range of b_i, $i \in \{0, 1, \ldots 63\}$ in the MPEG decoder [70] b_i, $\forall i \in \{0, 1, \ldots 63\}$ is limited to $-127 \leq b_i \leq 127$ for MPEG-1 and $-2048 \leq b_i \leq 2047$ for MPEG-2.

Let $\hat{D}$, defined from D as $\hat{d}_{i,j} = d_{8i+j}$, $i, j \in \{0, 1, \ldots 7\}$, denote the 8×8 matrix of pixel values after the IDCT.

In the MPEG decoder [70] $\hat{d}_{i,j}$ obtained from IDCT output is limited to $-127 \leq \hat{d}_{i,j} \leq 127, \forall i, j \in \{0, 1, \ldots 7\}$. If the result of calculations produces a value outside this range, it is set to -127 when $\hat{d}_{i,j} < -127$ and to 127 when $\hat{d}_{i,j} > 127$.

So we have,

$$\hat{D} = ((\hat{B} \cdot \hat{C})^T \cdot \hat{C})^T \tag{4.2}$$

which means that the two dimensional DCT is performed as two one dimensional DCTs, the first applied to each row and the second to each column.

4.2.2 Chosen DCT Coefficients Attack

A naive security evaluation of the system is performed by counting the number of possible permutations (size of the key space) and arguing that as long as finding the key by exhaustive search is infeasible, the scheme is secure. Tang [107] noted that if the attacker knows the plaintext corresponding to a ciphertext he can find the key. However no details of how such an attack would work, or experiments supporting this claim was presented. The systems proposed by Tang [107] and Shin *et al.*[95] are both based on random permutation and were both claimed to provide sufficient security.

We assume the following scenario. The attacker has obtained a decoder with a secret key set in the device, and aims to recover the key. He constructs a number of vectors of *attack DCT coefficients* and runs each vector through the decoder. By studying the output of the decoder, the attacker can find out the secret key. This attack scenario is called a *chosen-ciphertext* attack and is commonly used in security assessment of cryptosystems.

The attack exploits the fact that the structure of the MPEG stream is not hidden. In the permuted DCT scheme all data other than the DCT coefficients, including header parts, are not encrypted and so the attacker can modify the MPEG stream by replacing the permuted DCT coefficients with any value of his choice.

The basic idea is to construct a vector of 64 DCT coefficients which, after decoding, can reveal one or more *moves* of the original permutation. A move of P is defined by a pair (i, j) where i is the initial position of the element and j is the position after application of P.

Consider a vector S of 64 DCT coefficients such that there are n distinct values in the vector. If S is given to the decoder, the decoder applies the inverse permutation to find the original order of coefficients, and recover an image I' from the inverse-permuted coefficients. Next the attacker finds the DCT transform of I', and compares it with the original vector S. If the original n distinct coefficients can be unambiguously determined among the DCT coefficients of I', then n, $0 \leq n \leq 63$, *moves* of the inverse permutation Q (and hence in the permutation P) can be recovered.

An I-frame of MPEG-1 with $352 \times 240 \times 30$ frames per second (fps) defined in Source Input Format (SIF) contains 330 macro-blocks ($352/16 = 22, 240/16 = 15, 22 \times 15 = 330$) and $330 \times 6 = 1980$ blocks. If each block reveals the position of a single coefficient and the permutation is unchanged for $\lceil \frac{64}{n} \rceil$ blocks then it is possible to find the permutation.

Distinct DCT Coefficients

We can summarize the above procedure by starting from S, using the decoder on S which first calculates $Q(S)$ and then $I' = IDCT(Q(S))$. Now the attacker finds $DCT(I') = Q(S)$, and by comparing it with S, he can recover some moves of the permutation.

However because of inaccuracies resulting from DCT, IDCT and quantization, the $DCT(I')$ might be different from $Q(S)$ and so it might not be easy to distinguish all moves. To be able to accurately determine a move of P there must be no uncertainty about the coefficient values calculated in $DCT(I')$. This means that S must be chosen such that $|s_1 - s_2| \geq \epsilon,\ s_1, s_2 \in S$, ϵ a positive number, so that after all transformation steps, their identities remain distinguishable in $DCT(I')$.

Let $X = \{x_0, x_1, \ldots, x_{n-1}\}$ be a set of n distinct integers such that the minimum distance between two elements is bounded, that is, $m \leq |x_i - x_j|$ for all $x_i, x_j \in X$ where $i \neq j$.

Let an integer u be $u \notin X$ and $\min_{x_i} |u - x_i| \geq m$.

Then S for the attack coefficients vector is chosen such that $s_0 = x_0$, $s_1 = x_1$, ..., $s_{n-1} = x_{n-1}$, $s_n = u$, ..., $s_{63} = u$. That is, the DCT coefficients are such that n coefficients are $x_0, x_1, \ldots, x_{n-1}$ and $64 - n$ coefficients are u where $n < 63$.

Experiments

We conducted an experiment using an MPEG-2 coder program [70]. The first coefficient vector was, $B = (b_i)$, $b_0 = 64, b_i = -8, i \in \{1, 2, \ldots 63\}$. That is, $n = 1$, $x_0 = 64$ and $u = -8$. Other attack coefficients vectors were 63 non-identity permutations of the first vector. In each permutation the value "64" appeared in a different position : that is, 63 other vectors, $B^{(1)}, \cdots B^{(63)}$, where $B^{(j)} = (b^{(j)})$ were defined by $b_j^{(j)} = 64$, $b_i^{(j)} = -8$, $i \in \{0, 2, \ldots 63\}, i \neq j$. In the experiment, only I-frames were encrypted and the same permutation was used to encrypt a single I-frame. Each vector recovered a single move and to recover the entire permutation, only 63 blocks were needed. To reduce the effect of inaccuracy due to quantization we used the intra-quantization matrix table given by

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Using this matrix means that no scaling of the coefficients will be used (hence a considerable reduction in compression performance). In the chosen coefficient attack, the attacker can choose coefficients and the quantization matrix table as well.

Discussion

We have shown that as long as the permutation remains unchanged for 64 blocks the random permutation can be revealed. Optimizing the attack allows recovery of more than one move per attack DCT coefficient vector and effectively reduces the number of required attack coefficient vectors. Finding the least number of such vectors gives a bound on security of the system.

In the scheme by [107], an added security measure is the encryption of blocks of 8 DC coefficients using DES (Data Encryption Standard [38]). Finding DC coefficients

therefore requires finding the DES key or finding plaintext for a DES encrypted block which is well studied in the literature (such as [66]). However correctly recovered AC coefficients could reveal edge information in an image as shown in Figure 7 in the paper [107]. It is also worth noting that this is a relatively expensive system as for 8 image blocks ($8 \times 64 = 512$ pixels) one DES encryption is required. For example, for 352×240 frame size and one I-frame / sec MPEG stream, at least $1980/8 \approx 247$ DES blocks need to be encrypted in every second. In [95], the added security measure is encrypting the sign bits of DC coefficients using a RC4 [85, 86] stream cipher. That is, the sign bits are XORed with the output of RC4. However the attack described in this chapter can recover the sign bits by simply XORing the sign bits of the original DC coefficients and the ones obtained by the attack. This is because the key for the RC4 stream cipher is chosen independently from the input MPEG data and so the same key is used for the attack coefficient vectors. This means that this system is completely insecure under the above chosen attack coefficients (the attack coefficients must be chosen such that the DC coefficient is non-zero).

Another proposed technique for improving security has been splitting DC coefficients which again will become ineffective when the permutation is found and the DC coefficient is reassembled. The above analysis suggests extra measures, such as hiding quantization table or structure of MPEG stream, must be used to provide reasonable security.

4.2.3 Concluding Remarks

As shown above, chosen DCT coefficients attack can discover not only the random permutation used in the coder but also the encryption of sign-bits. If the permutations and the encryption of sign-bits are known, the attacker is able to recover the encrypted stream.

4.3 Recovering the DC Coefficient in Block-based Discrete Cosine Transform

MPEG encryption systems [107, 95] apply a secret permutation to DCT coefficients. However in each block, the DC coefficient carries most of the signal energy and is much larger than the other coefficients and so it is easily distinguishable after permutation. If the DC coefficients of blocks are known, a low resolution version of the image can

be constructed. This means that permutation of the AC coefficients cannot make the image incomprehensible. The two proposed methods [107, 95] use a strong encryption algorithm to encrypt the DC coefficients of blocks while permuting the other coefficients. The method is claimed to produce images that are *incomprehensible*, and so the scrambled images do not leak any information to outsiders.

We examine whether or not it is possible to recover the DC coefficients of image blocks when only the AC coefficients are known and then find the quality level of the recovered images. In Section 4.3.1, we review properties of DCT that are useful for our attack. In Section 4.3.2, the DC recovery attack is developed and Section 4.3.3 reports on the results of experiments. Finally we conclude our results.

4.3.1 Properties of DCT Coefficients

Block-based DCT

Let us consider an $N \times N$ image block $[X]$, where x_{ij} is the value of the pixels in the ith row and the jth column. Further, let the maximum possible value of a pixel in the image be $x_{\max}$ and the smallest quantization increment of a pixel value be $x_{\min}$. The DCT of the pixel block is given by the matrix $[C]$,

$$[C] = [A][X][A]^t \tag{4.3}$$

where $[A]$ is the matrix of DCT basis vectors. The image block is recovered through the inverse transform as,

$$[X] = [A][C][A]^t \ . \tag{4.4}$$

Let $[C]^{DC}$ and $[C]^{AC}$ denote $N \times N$ matrices of DCT coefficients where all AC coefficients in $[C]^{DC}$ are zeros and the DC coefficient in $[C]^{AC}$ is zero, respectively. Let $[X]^{DC}$ and $[X]^{AC}$ be $[X]^{DC} = [A][C]^{DC}[A]^t$ and $[X]^{AC} = [A][C]^{AC}[A]^t$, respectively. Then for a given image block k, the pixel values can be written as a decomposition into a DC and an AC component. Thus,

$$[X]_k = [X]_k^{DC} + [X]_k^{AC} \ . \tag{4.5}$$

Let $F_k^{(i,j)}$ be a DCT coefficient of a block K at (i,j) position. Of course, $[X]_k^{DC}$ is constant over all $i, j \in N$ and its components are given by,

$$(x^{DC})_k^{(i,j)} = A * F_k^{(1,1)}, \text{ where } A = \frac{1}{N^2} \ . \tag{4.6}$$

Hence in the case of JPEG where the block size is 8×8, the multiplying factor is $\frac{1}{64}$. The dynamic range of the DC coefficients has been shown [16] to be

$$\Lambda_{DC} = \frac{x_{\max}}{x_{\min}} \cdot N^2 \ . \tag{4.7}$$

The AC coefficients can have both positive and negative values with dynamic range given by,

$$\Lambda_{AC}(i,j) = \frac{x_{\max}}{x_{\min}} \cdot \frac{\sum_{n=1}^{N} |F^{(i,n)}| \sum_{n=1}^{N} |F^{(j,n)}|}{|F^{(i,n)}_{(\min)}||F^{(j,n)}_{(\min)}|}) \ . \tag{4.8}$$

The dynamic range of the AC coefficients varies from coefficient to coefficient.

Relationship between DC Coefficients of Neighboring Blocks

The DC coefficient of an image block represent the mean value of pixels in the image block. These DC values can construct a decimated (and low-pass filtered) version of the original image. For a natural image, the pixel level correlation structure is carried over to the low-pass filtered version of the image. In the smooth areas of the image, the prediction error of a first-order predictor is small and large prediction errors are observed around the edges. In general, the values of pixels in $[X]_k^{AC}$ has been modeled as a zero-mean Laplacian distributed random variable, $p(x) = \frac{\lambda}{2} e^{-\lambda|x|}$, whose distribution generally has a small variance.

As shown in equation (4.5), the pixel values can be decomposed into DC and AC parts. The DC part is constant while the AC part is the mean-removed pixel value. A large dynamic range for the AC part constrains the DC part to a small value since the total dynamic range cannot exceed that of the pixels. We summarize these considerations into two properties.

Property 1 The difference between two neighboring pixels is a Laplacian random variable with zero mean and small variance.

Property 2 The dynamic range of $[X]^{AC}$ constrains the value of $[X]^{DC}$ because $0 \leq x^{(i,j)} \leq x_{\max}, x^{(i,j)} \in [X]$ and $[X] = [X]^{DC} + [X]^{AC}$. In particular large dynamic range for $[X]^{AC}$ implies small values for $[X]^{DC}$. The converse is also true.

A pixel is in the neighborhood of pixel x_{ij} if it belongs to $\Phi = \{x_{(i+1)(j-1)}, x_{(i+1)j}, x_{(i+1)(j+1)}, x_{i(j+1)}, x_{i(j-1)}, x_{(i-1)(j-1)}, x_{(i-1)j}, x_{(i-1)(j+1)}\}$.

Let $T^{(k)}$ and $T^{(k+1)}$ denote two adjacent 8×8 pixel blocks, with pixel values given by $x_{ij}^{(k)}$ and $x_{ij}^{(k+1)}$, respectively.

$$T^{(k)} = \begin{pmatrix} x_{11}^{(k)} & x_{21}^{(k)} & \cdots & x_{81}^{(k)} \\ x_{12}^{(k)} & x_{22}^{(k)} & \cdots & x_{82}^{(k)} \\ \vdots & \vdots & \vdots & \vdots \\ x_{18}^{(k)} & x_{28}^{(k)} & \cdots & x_{88}^{(k)} \end{pmatrix}$$

$$T^{(k+1)} = \begin{pmatrix} x_{11}^{(k+1)} & x_{21}^{(k+1)} & \cdots & x_{81}^{(k+1)} \\ x_{12}^{(k+1)} & x_{22}^{(k+1)} & \cdots & x_{82}^{(k+1)} \\ \vdots & \vdots & \vdots & \vdots \\ x_{18}^{(k+1)} & x_{28}^{(k+1)} & \cdots & x_{88}^{(k+1)} \end{pmatrix} .$$

The two blocks can be horizontally or vertically adjacent. Given the distribution of the difference of two adjacent pixels, the difference between the two *neighboring* pixels, is likely to be zero. Let $(p^{(k)}, p^{(k+1)})$ denote a pair of neighboring pixels, where $p^{(k)}$ is in $T^{(k)}$ and $p^{(k+1)}$ is in $T^{(k+1)}$.

The two pixels can be related as,

$$p^{(k)} = p^{(k+1)} + \varepsilon \tag{4.9}$$

where ε has zero mean. In the following we will show that when (4.9) holds, the DC signals of the two blocks can be related through the values of the AC signals in the two blocks.

Assume the coefficients, $F_k^{(i,j)}, i, j \in \{1, 2, \ldots, 8\}$ and $F_{k+1}^{(i,j)}, i, j \in \{1, 2, \ldots, 8\}$ are known, except for the DC coefficient, $F_{k+1}^{(1,1)}$ of block $k+1$. That is, x_k^{AC}, x_k^{DC} and x_{k+1}^{AC} are known but x_{k+1}^{DC} is unknown. We apply the inverse-transform to the AC coefficients of the two blocks to obtain the matrix of DC-free pixels values, $\Gamma^{(k)}$ and $\Gamma^{(k+1)}$, corresponding to X_k^{AC} and X_{k+1}^{AC}, respectively.

Let $(\rho^{(k)}, \rho^{(k+1)})$ denote a pair of neighboring DC-free pixel values of the pair $(p^{(k)}, p^{(k+1)})$.

From equation (4.5) and (4.6), we have :

$$\begin{aligned} p^{(k)} &= \rho^{(k)} + A \cdot c_{11}^{(k)} \\ p^{(k+1)} &= \rho^{(k+1)} + A \cdot c_{11}^{(k+1)} . \end{aligned} \tag{4.10}$$

From equation (4.9),

$$\rho^{(k)} + A \cdot c_{11}^{(k)} \approx \rho^{(k+1)} + A \cdot c_{11}^{(k+1)}$$

$$c_{11}^{(k+1)} \approx \frac{\rho^{(k)} - \rho^{(k+1)}}{A} + c_{11}^{(k)} \ . \tag{4.11}$$

Hence, since $\rho^{(k)}$, $\rho^{(k+1)}$ and $c_{11}^{(k)}$ are known, the value of $c_{11}^{(k+1)}$ can be found.

4.3.2 Recovering the DC Coefficients in a Block-based DCT

Estimating the DC Coefficient of a Block

The relationship (4.11) between the DC coefficients of two neighboring blocks holds only if we know a pair of neighboring pixels that satisfy (4.9). If the actual pixel values of adjacent blocks are unknown but their DC-free values are given, we can use equation (4.11) to estimate the DC value of one block in terms of its neighbors.

Consider DC-free values of pixels in two adjacent blocks. We use equation (4.11) on pairs of horizontally neighboring pixels to obtain estimates of $c_{11}^{(j+1)}$ from pairs $(\rho_i^{(j)}, \rho_i^{(j+1)}), i \in \{1, \ldots, 8\}$, and then obtain a final estimate as the average of all such estimates. The following diagram shows the DC-free pixel values of two columns of two adjacent blocks.

$\ldots$	$\rho_1^{(j)}$	$\rho_1^{(j+1)}$	$\ldots$
$\ldots$	$\rho_2^{(j)}$	$\rho_2^{(j+1)}$	$\ldots$
$\ldots$	$\vdots$	$\vdots$	$\ldots$
$\ldots$	$\rho_8^{(j)}$	$\rho_8^{(j+1)}$	$\ldots$

Suppose the jth block is the reference block and let $\Delta_{j+1} = c_{11}^{(j+1)} - c_{11}^{(j)}$ denote the estimated adjustment for pixels in the $j+1$th block. Then, we use

$$\Delta_{j+1} = \frac{\sum_{i=1}^{8} (\rho_i^{(j)} - \rho_i^{(j+1)})}{8 \cdot A} \tag{4.12}$$

as the final estimate of the difference and the corrected pixel values $\rho'^{(j+1)}$ are calculated as

$$\rho'^{(j+1)} = \rho^{(j+1)} + A \cdot \Delta_{j+1} \ . \tag{4.13}$$

The noise induced by this estimation can be reduced by taking an average over a number of pixels. This method will perform well only if condition (4.9) is satisfied for most of the pixel pairs used in the estimation.

We only considered horizontal and vertical neighbors. If there is a horizontal line across an image, the difference between two neighboring pixels along the line will be zero but for two vertically neighboring pixels, one on the line and the other next to

the line, the difference will not be zero and so the estimates in equation (4.12) will be poor. In the following section, we show how to improve the accuracy of estimation in the above situation.

Improving the Algorithm

Estimating the DC value as above works well if horizontally (vertically) neighboring pixels in two adjacent columns (rows) have close values. For regions such as Lena's hat in Figure 4.1, with high variation in horizontal and vertical directions but smooth in diagonal directions, the algorithm will produce poor estimates. In the following we modify the algorithm to find the *smoothest* direction among the horizontal, vertical and the two diagonal directions, and use it to find an estimate of the DC value.

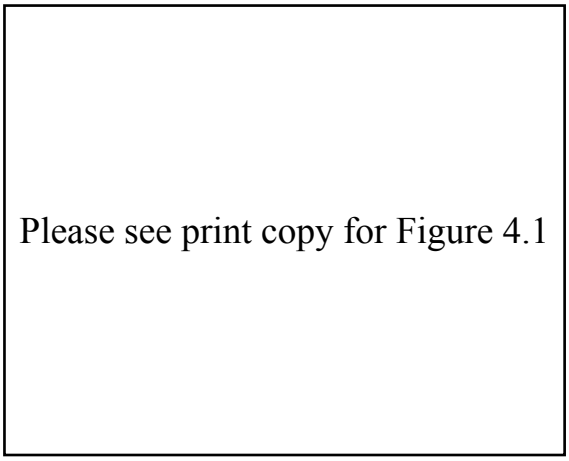


Figure 4.1: Gray scale Lena picture.

The basic idea is to consider three sets of pixel pairs in two adjacent columns (rows) that correspond to horizontal (vertical) and two diagonal directions, and use the mean square error to choose the smoothest direction.

Let $T^{(k)}$ be the block adjacent to $T^{(j)}$ in horizontal (vertical) direction. (For two horizontally adjacent blocks $k = j + 1$ and for two vertically adjacent ones $k = j + l$ where l is the number of blocks in one row.) The three sets of pixels are: *i*) $(\rho_i^{(j)}, \rho_i^{(k)})$ where $1 \leq i \leq 8$ (pattern 1 in Figure 4.2), *ii*) $(\rho_{i+1}^{(j)}, \rho_i^{(k)})$ where $1 \leq i \leq 7$ (pattern 2 in Figure 4.2), and *iii*) $(\rho_i^{(j)}, \rho_{i+1}^{(k)})$ where i, $1 \leq i \leq 7$ (pattern 3 in Figure 4.2). Then the smoothest direction among the three possibilities is chosen to estimate the DC value.

Consider 3 vectors consisting of pixels in $\Gamma^{(j)}$ and 3 vectors of pixels for $\Gamma^{(k)}$ as

$$\begin{aligned} \beta_1^{(j)} &= (\rho_1^{(j)}, \rho_2^{(j)}, \ldots, \rho_8^{(j)}) \\ \beta_2^{(j)} &= (\rho_2^{(j)}, \rho_3^{(j)}, \ldots, \rho_8^{(j)}) \end{aligned}$$

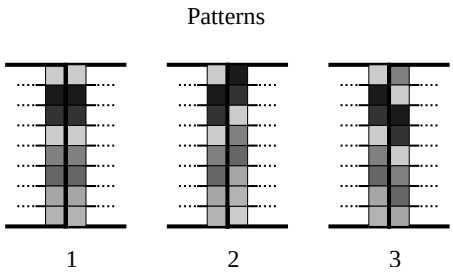


Figure 4.2: Possible pixels patterns at the border in the case of a pair of horizontally neighboring blocks.

$$\begin{aligned}
\beta_3^{(j)} &= (\rho_1^{(j)}, \rho_2^{(j)}, \ldots, \rho_7^{(j)}) \\
&\text{and} \\
\beta_1^{(k)} &= (\rho_1^{(k)}, \rho_2^{(k)}, \ldots, \rho_8^{(k)}) \\
\beta_2^{(k)} &= (\rho_1^{(k)}, \rho_2^{(k)}, \ldots, \rho_7^{(k)}) \\
\beta_3^{(k)} &= (\rho_2^{(k)}, \rho_3^{(k)}, \ldots, \rho_8^{(k)}) \ .
\end{aligned} \tag{4.14}$$

The above 3 sets of pairs are $\beta_v^{(j)}$ and $\beta_v^{(k)}$, where $v = 1, 2, 3$.

Then the algorithm to calculate the DC is as follows.

1. Calculate the means, $M_v^{(j)}$ and $M_v^{(k)}$, of $\beta_v^{(j)}$ and $\beta_v^{(k)}$ as follows.

$$\begin{aligned}
M_1^{(j)} &= \frac{\Sigma_{n=1}^{8} \rho_n^{(j)}}{8} \\
M_2^{(j)} &= \frac{\Sigma_{n=2}^{8} \rho_n^{(j)}}{7} \\
M_3^{(j)} &= \frac{\Sigma_{n=1}^{7} \rho_n^{(j)}}{7} \\
&\text{and} \\
M_1^{(k)} &= \frac{\Sigma_{n=1}^{8} \rho_n^{(k)}}{8} \\
M_2^{(k)} &= \frac{\Sigma_{n=2}^{8} \rho_n^{(k)}}{7} \\
M_3^{(k)} &= \frac{\Sigma_{n=1}^{7} \rho_n^{(k)}}{7} \ .
\end{aligned} \tag{4.15}$$

2. Subtract the mean $M_v^{(j)}$ from the vector $\beta_v^{(j)}$ and $M_v^{(k)}$ from $\beta_v^{(k)}$.

$$\begin{aligned}
\hat{\beta}_1^{(j)} &= (\rho_1^{(j)} - M_1^{(j)}, \ldots, \rho_8^{(j)} - M_1^{(j)}) \\
\hat{\beta}_2^{(j)} &= (\rho_2^{(j)} - M_2^{(j)}, \ldots, \rho_8^{(j)} - M_2^{(j)}) \\
\hat{\beta}_3^{(j)} &= (\rho_1^{(j)} - M_3^{(j)}, \ldots, \rho_7^{(j)} - M_3^{(j)}) \\
&\text{and} \\
\hat{\beta}_1^{(k)} &= (\rho_1^{(k)} - M_1^{(k)}, \ldots, \rho_8^{(k)} - M_1^{(k)})
\end{aligned}$$

$$\begin{aligned} \hat{\beta}_2^{(k)} &= (\rho_2^{(k)} - M_2^{(k)}, \ldots, \rho_8^{(k)} - M_2^{(k)}) \\ \hat{\beta}_3^{(k)} &= (\rho_1^{(k)} - M_3^{(k)}, \ldots, \rho_7^{(k)} - M_3^{(k)}) \ . \end{aligned} \tag{4.16}$$

3. Calculate the mean square difference of $\hat{\beta}_v^{(j)}$ and $\hat{\beta}_v^{(k)}$ as follows.

$$\Omega_v = (1/t)(\hat{\beta}_v^{(j)} - \hat{\beta}_v^{(k)})^2 \ . \tag{4.17}$$

   where $t = 7$ when $v = 1$, and $t = 8$ otherwise.

4. Find $\min_v \Omega_v$ and the corresponding $\beta_v^{(j)}$ and $\beta_v^{(k)}$.

5. Assuming that the j th block is the reference block and the pixels in the k th block are adjusted, the adjustment value Δ_k is given by

$$\Delta_k = \frac{M_v^{(j)} - M_v^{(k)}}{A} \ . \tag{4.18}$$

   and the new pixel values $\rho_i^{\prime(k)}$ are calculated as

$$\rho_i^{\prime(k)} = \rho_i^{(k)} + A \cdot \Delta_k \ . \tag{4.19}$$

Bounding the DC Value of a Block

Property 2 can be used to bound the dynamic range $\Lambda^{(j)}$ of the DC coefficient of a block. Let the pixels $\rho^{(j)}$ in block $T^{(j)}$ have the range,

$$\lambda_{\min}^{(j)} \leq \rho^{(j)} \leq \lambda_{\max}^{(j)} \tag{4.20}$$

and assume the possible values of pixels are in the interval,

$$0 \leq p_i^{(j)} \leq t_{\max}, \forall j, i \ . \tag{4.21}$$

Then the following must hold.

$$0 \leq \rho^{(j)} + A \cdot c_{11}^{(j)} \leq t_{\max} \ . \tag{4.22}$$

From equation (4.20) and (4.22),

$$0 \leq \lambda_{\max}^{(j)} + A \cdot c_{11}^{(j)} \tag{4.23}$$

and

$$\lambda_{\min}^{(j)} + A \cdot c_{11}^{(j)} \leq t_{\max} \ . \tag{4.24}$$

Hence

$$\frac{\lambda_{\max}^{(i)}}{A} \leq c_{11}^{(j)} \leq \frac{t_{\max} - \lambda_{\min}^{(i)}}{A} \ . \tag{4.25}$$

Recovering the DC Value of the Image

Using the result of Sections 4.3.2 and 4.3.2 we describe an algorithm that recovers DC signal of blocks. The two steps of the algorithm, that is estimating relative values of DC signals and then estimating the actual DC signal, are described in the following two sections.

Adjusting Relative Values of the DC Signals

We use the methods described in Section *Estimating the DC coefficient of a block* (p.64) to estimate the relative DC signals of blocks in an image in terms of their adjacent blocks. If the DC signals of all blocks are unknown, then without loss of generality we assume the top left block in the image is the reference block. The range of the DC signal for the block can be obtained from equation (4.25). We calculate the DC signals of all other blocks in terms of the DC signal of the reference block.

We note that to use the algorithm in Section *Estimating the DC coefficient of a block* to find an estimate for $[X]_j^{DC}$, we may choose one of the 4 possible adjacent blocks. This means that to cover all blocks in the image starting from a reference block, various paths through the image blocks can be considered.

As noted earlier, to estimate the DC value of a block, one or more of its neighboring blocks can be used. The algorithm below is an example of systematically adjusting all blocks of an image.

1. First pass
   (a) The block in the upper left corner of the image is chosen as the reference block. Blocks in the first row are considered from left to right, and in each case its DC value is adjusted with respect to its left block.
   (b) The rows below the first are adjusted similarly to the above but each block, except the left-most ones, is compared to its upper and left blocks and is adjusted based on the average of the 2 estimated adjustment values. For a left-most block, only its upper block is considered.
2. Second pass
   (a) The block in the upper right corner of the image with its DC value adjusted in the first pass, is chosen as the reference block. The first row of blocks is adjusted from right to left. The DC of each block is calculated relative to its left block.

(b) The rows below the first are adjusted similarly to the above but each block, except the right-most ones, is compared with its upper and right blocks and is adjusted based on the average of the 2 adjustment values. For a right-most block only its upper block is considered.

3. Third pass

(a) The block in the bottom left corner of the image with its DC value adjusted in the second pass, is chosen as the reference block. Then the blocks at the bottom row are considered from left to right. The DC of each block is calculated with reference to its left block.

(b) The rows above the bottom row are adjusted similarly to the above, but each block, except the left-most ones, is compared with its lower and left blocks and is adjusted using the average of the two adjustment values. For a left-most block only its lower block is considered.

4. Fourth pass

(a) The block in the bottom right corner of the image with its DC value adjusted in the third pass, is chosen as the reference block. The first row of blocks is adjusted from right to left. The DC of each block is calculated with reference to its left block.

(b) The rows above the bottom are adjusted similarly but each block, except the right-most ones, is compared with its lower and right blocks and is adjusted based on the average of the two adjustment values. For a right-most block only its lower block is considered.

Adjustment of the Pixel Dynamic Range

After the relative adjustment of the DC values of all blocks, it is necessary to find the actual values of the DC signal for the entire image. The adjustment in the previous section does not take into account possible range of pixels in a block and so during the adjustment some pixels in the image may move outside the valid pixel range.

The range of DC signal in each block can be obtained from equation (4.25). The effective range of $c_{11}^{(j)}$, the DC value of the reference block, is the smallest range of all blocks. This is because changing $c_{11}^{(1)}$ from 0 to $\alpha > 0$, adds the same value to all $c_{11}^{(j)}$ for all $j = 2, \ldots,$ and the new value of $c_{11}^{(j)}$ must stay within the dynamic range, $\Lambda^{(j)}$.

The dynamic range of the pixels may be larger than the valid pixel range due to the inaccuracy in the recovery. To fit all pixel values within the valid range, either all pixel values must be scaled or only the pixel values outside the valid range must be adjusted. The exact value of $[X]_1^{DC}$ will be determined in a subjective way and by examining the quality of the resulting image.

4.3.3 Experiment Results

In this section we show the distribution of differences of neighboring pixels and the results of DC recovery experiments. For our experiments, we used four gray scale images, `airfield256x256.pgm` (256×256 pixels), `mandoril.pgm` (512×512 pixels), `lena.pgm` (512×512 pixels), and `peppers.pgm` (512×512 pixels).

Distribution of Differences of Pixels

The list below shows the means and standard deviations of differences of neighboring pixels and the pixel value ranges in the images. Figure 4.3 shows the distribution of the differences of neighboring pixels. From the results, it can be seen that the distribution of the differences were a zero-mean Laplacian.

Image	Mean	Std. dev	Pixel range
`airfield256x256.pgm`	0.04	33.9	0 - 255
`mandoril.pgm`	-0.18	34.9	0 - 255
`lena.pgm`	0.01	11.5	24 - 245
`peppers.pgm`	-0.35	19.5	0 - 225

DC Recovery Experiments

We used the algorithms described in Section 4.3.2 to recover the images whose DCT coefficients, excluding the DC coefficient, are given. The steps used for the experiments were as follows.

1. Transform the image using the 8×8 two dimensional DCT.
2. All the DC coefficients are set to 1023, which is the middle value of the dynamic range of DC.
3. The methods described in Section *Estimating the DC coefficient of a block* , *Improving the algorithm* , *Adjusting relative values of the DC signals* and *Adjustment of the pixel dynamic range* , were used to recover the DC values.

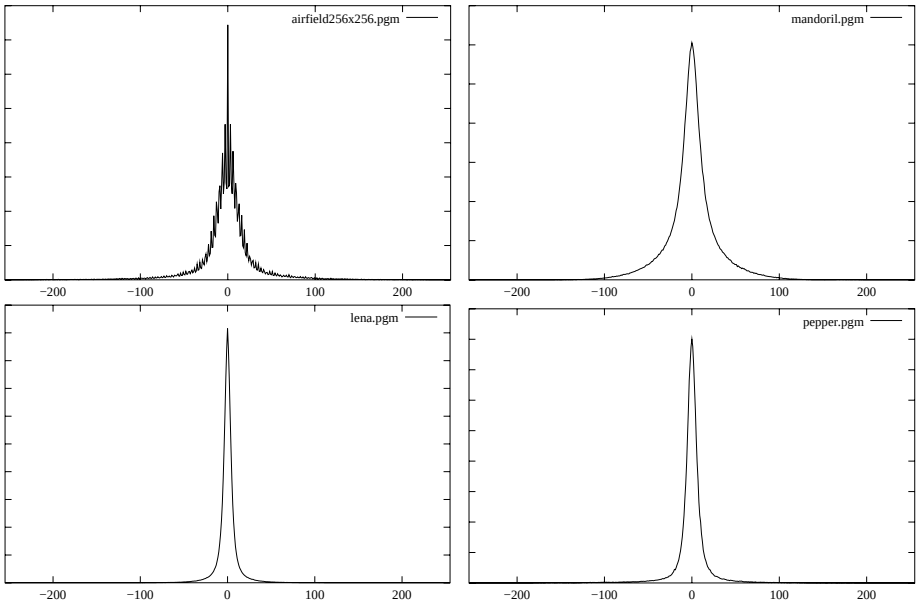


Figure 4.3: The distribution of differences of neighboring pixels in `airfield256x256.pgm` (left top), `mandoril.pgm` (right top), `lena.pgm` (left bottom), and `peppers.pgm` (right bottom).

DC Recovery When All DC Coefficients Are Unknown

Table 4.1, Figure 4.4, Table 4.2, and Figure 4.5 summarize the recovery results for the two algorithms described in Section *Estimating the DC coefficient of a block* and *Improving the algorithm* .

Table 4.1: Quality of the recovered images using the method in Section *Estimating the DC coefficient of a block* .

Image	PSNR
airfield256x256.pgm	23.1 dB
mandoril.pgm	18.2 dB
lena.pgm	22.9 dB
peppers.pgm	17.7 dB

DC Recovery When Some of the DC Coefficients Are Known

In the row direction, the DC values of the odd position blocks were set to 1023, and the even position blocks keep the original DC values. Hence half of all blocks have their DC coefficients destroyed. The steps used for the experiment were as follows.

1. Transform the image using an 8×8 two dimensional DCT.

2. Half of the DC coefficients are set to 1023.

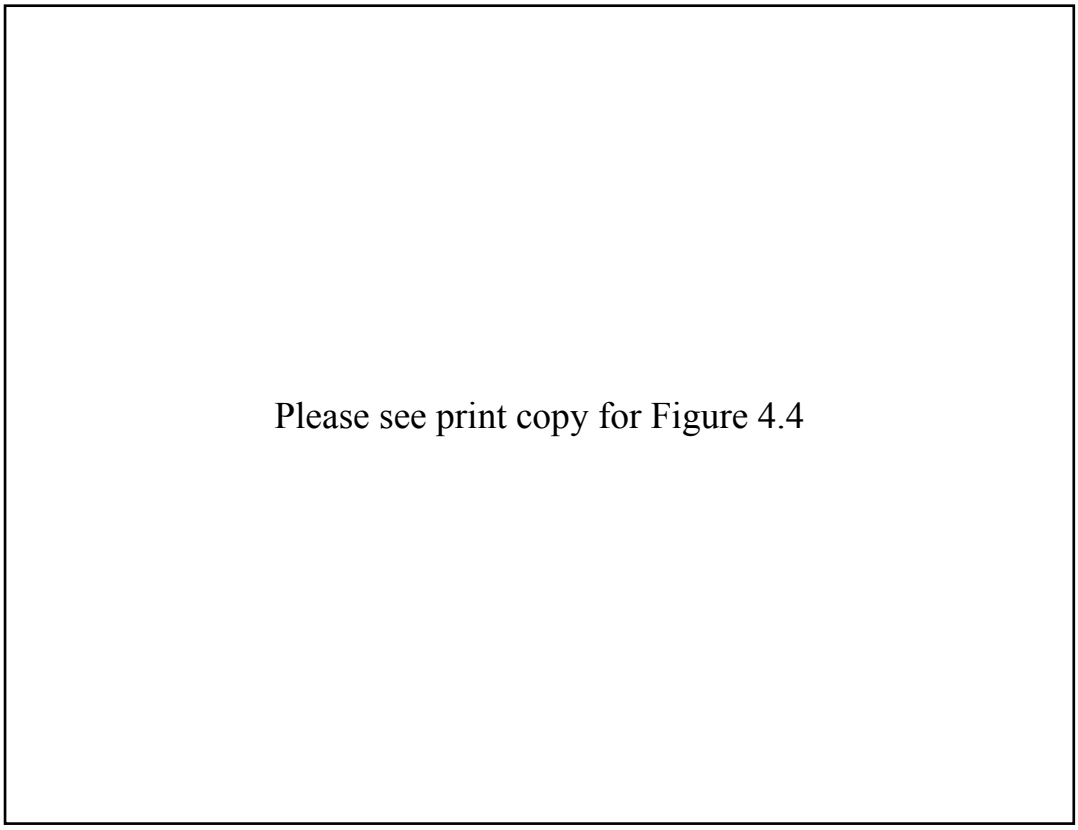


Figure 4.4: The images recovered by the method in Section *Estimating the DC coefficient of a block* . `airfield256x256.pgm` (top left), `mandrill.pgm` (top right), `lena.pgm` (bottom left) and `peppers.pgm` (bottom right).

3. The methods in Section *Improving the algorithm* , *Adjusting relative values of the DC signals* and *Adjustment of the pixel dynamic range* , were used to recover the DC values.

Table 4.3 and Figure 4.6 show the recovery results with half of the DC coefficients of the images.

4.3.4 Another Application of DC Recovery

DC recovery can be used to reduce the number of DC coefficients that are encoded when an image is compressed. Since DC coefficients can be recovered from AC coefficients, it is not necessary to encode all DC coefficients. This can be used to reduce the size of compressed data or to embed information in the image by encoding data instead of DC coefficients.

Since encoding no DC coefficient would result in poor image quality, the number of DC coefficients that are encoded can be chosen according to the required image quality. If half of the DC coefficients of four images, `airfield256x256.pgm mandoril.pgm`

Table 4.2: Image quality of the recovered images using the method in Section *Improving the algorithm* .

Image	PSNR
airfield256x256.pgm	23.5 dB
mandoril.pgm	17.7 dB
lena.pgm	23.8 dB
peppers.pgm	18.6 dB

Table 4.3: Quality of the recovered images with half of the DC coefficients in the image.

Image	PSNR
airfield256x256.pgm	32.1 dB
mandoril.pgm	31.0 dB
lena.pgm	37.1 dB
peppers.pgm	34.4 dB

`lena.pgm` and `peppers.pgm` are encoded, the recovered images and their PSNRs are shown in Figure 4.6 and Table 4.3.

Table 4.4, 4.5, and 4.6 show the size of the entropy-coded DC coefficients of the images `airfield256x256.pgm` `mandoril.pgm` `lena.pgm` and `peppers.pgm` To obtain the data, the command `cjpeg` [111] was used with the quality level 50%, 75% and 90% with the default quantization table. The *DC size* shows the size of entropy-coded DC coefficients (i.e. the difference of a DC coefficient from the previous one) using the Huffman coder and the ratio of the size of encoded DC coefficients to the file size in percent. The DC size in the table shows the maximum size to be reduced using DC recovery.

To recover DC coefficients, the decoder needs to obtain all AC coefficients. If AC coefficients in a block are lost (for example, due to transmission errors), the estimation of DC coefficients in the block will be inaccurate and so the recovered image will have low quality.

Table 4.4: The sizes of the JPEG file and encoded differential DC values in the file for image quality=50%.

Image	JPEG size (bytes)	DC size (bytes)	Ratio of DC in the file
airfield256x256.pgm	12935	807.5	6.2%
mandoril.pgm	50836	2905.375	5.7%
lena.pgm	20918	2959.875	14.1%
peppers.pgm	8072	829.875	10.3%

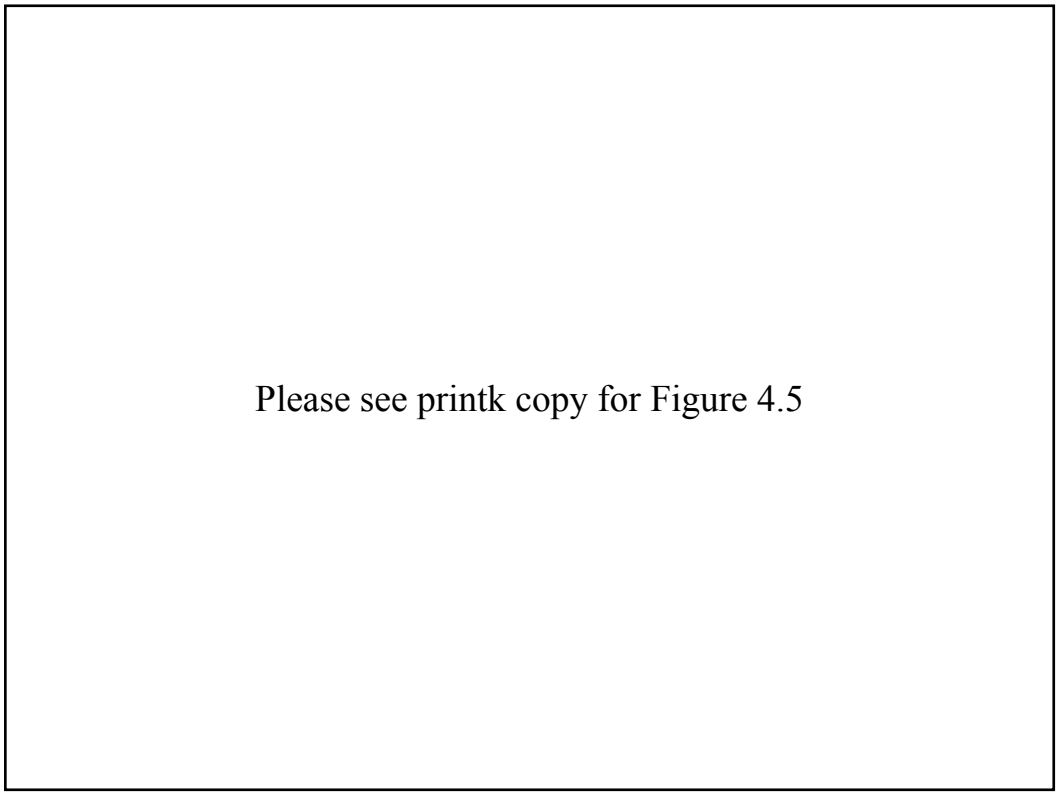


Figure 4.5: The images recovered by the method in Section *Improving the algorithm* . `airfield256x256` (top left), `mandrill` (top right), `lena` (bottom left) and `peppers` (bottom right).

4.3.5 Concluding Remarks

We showed that if block based DCT is used on images, then it is possible to find an estimate of the DC signal of a block from the AC signal of that block and the complete signal of its neighboring blocks. The method selects the smoothest direction of natural images and only considers horizontal, vertical or diagonal direction. It is possible to increase the number of directions, for example, using every 5 degree direction, to obtain a more precise direction of smoothness and hence a better estimate of the DC signal.

An application of the results of this section is a new attack on DCT encryption systems. It has been argued that DCT encryption systems that use permutation of the AC coefficients together with encryption of the DC coefficients provide high security and result in incomprehensible images. Using the attack in Section 4.2 together with the results in this section shows that the claimed level of security of the systems [107, 95] does not hold.

Another interesting application of the results is that in JPEG it is not necessary to encode DC signals of all blocks. Rather it is sufficient to encode the DC signal of some of the blocks and in the recovery phase, use methods similar to those described in this

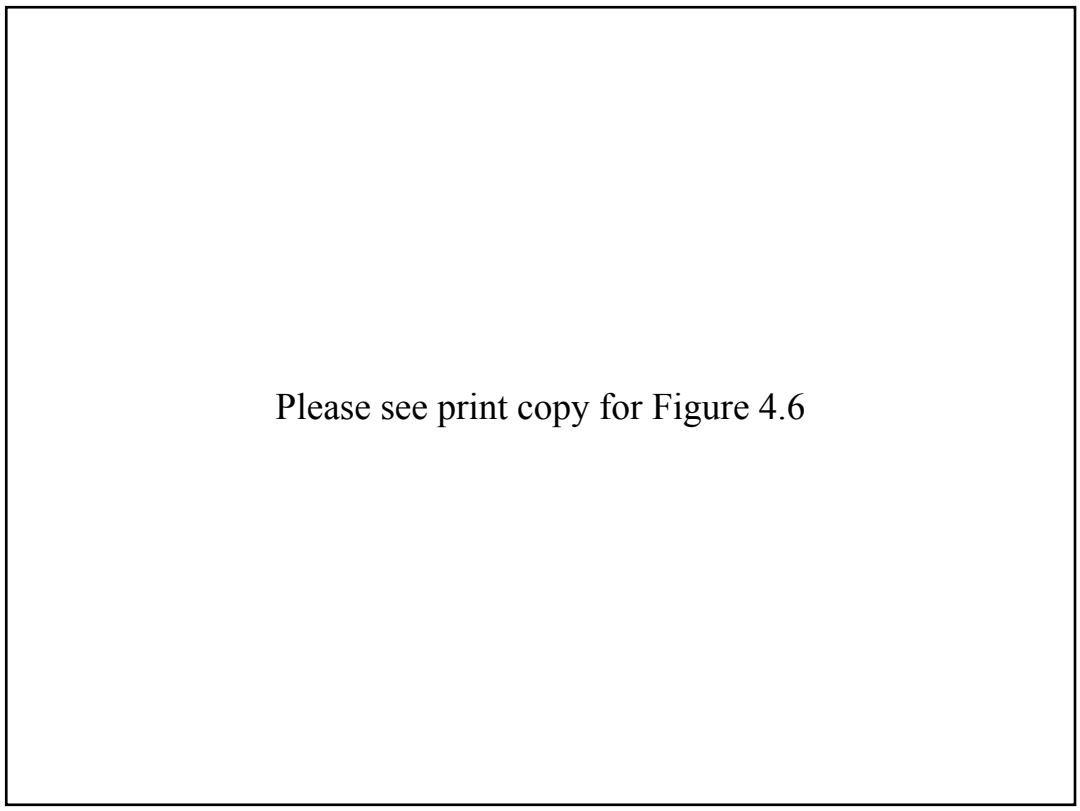


Figure 4.6: The images recovered from the half of DC signals by the method in *Improving the algorithm* . `airfield256x256.pgm` (top left), `mandrill.pgm` (top right), `lena.pgm` (bottom left) and `peppers.pgm` (bottom right).

section to find the remaining ones. This results in some loss of quality, but a higher compression ratio at the cost of increased computation for decoding. For example, encoding only half of the DC coefficients results in a 37 dB image quality (Section 4.3.3). The trade-off between the quality of the recovered image and the required computation, and also the theoretical limit of the quality of the recovered image are interesting open problems.

4.4 Conclusion

We have shown new attacks on MPEG encryption systems which can be also used on JPEG encryption systems. The chosen DCT coefficients attack is able to find permutations of coefficients and so MPEG and JPEG encryption systems using random permutation lists for encryption are vulnerable against the attack. We also have shown that hiding only the DC coefficients does not provide security although it will largely degrade image quality.

For the above two attacks to be successful, it is necessary for the attacker to obtain

Table 4.5: The sizes of the JPEG file and encoded differential DC values in the file for image quality=75%.

Image	JPEG size (bytes)	DC size (bytes)	Ratio of DC in the file
airfield256x256.pgm	19819	957.25	4.8%
mandoril.pgm	82376	3396.75	4.1%
lena.pgm	32570	3520	10.8%
peppers.pgm	12222	984.5	8.1%

Table 4.6: The sizes of the JPEG file and encoded differential DC values in the file for image quality=90%.

Image	JPEG size (bytes)	DC size (bytes)	Ratio of DC in the file
airfield256x256.pgm	32130	1218.75	3.8%
mandoril.pgm	128401	4320.5	3.4%
lena.pgm	59203	4453	7.5%
peppers.pgm	21157	1249.125	5.9%

frames which the decoder outputs. To avoid the attacks, hiding the parameters and the structure of a MPEG stream can be used as an additional encryption because the decoder fails to synchronize the stream if the parameters and the structure are hidden and so it will not produce any output.

Chapter 5

JPEG Encryption

5.1 Introduction

Secure distribution of information is crucial in multimedia applications. Multimedia data are mostly in compressed form. Combining security and compression can increase system efficiency. The challenge is to provide security without significant drop in the compression rate or large increase in computational cost.

The objective of this chapter is to propose an efficient encryption system for image data such that *i*) compression drop is negligible, *ii*) high level of security is obtained, and *iii*) the encrypted data conforms to the JPEG image compression standard specification.

Using a computationally expensive encryption algorithm is not acceptable in many applications. To achieve efficient encryption for image data, there are two known approaches : *i*) using computationally inexpensive *primitive cryptographic operations* on the whole stream, and *ii*) *selective encryption* which only encrypts selected part of the stream instead of encrypting the whole stream. Encryption schemes using primitive cryptographic operations for MPEG [107, 95] are shown to be weak against known plaintext and chosen ciphertext attacks described by Agi et al. [2, 114] and Chapter 4.

If the encrypted stream remains conformant to the data format specified by the JPEG specification, information which is not encrypted, for example the size of an image, can be obtained without decryption. For example, if a web page includes an image, a web browser needs to know the size of the image to show the page in the correct layout. If the encrypted stream remains conformant to the JPEG specification, without decryption a web browser can display an encrypted image, which has the correct size but is visually corrupted.

We propose a scheme that avoids high computation cost and uses selective encryption. JPEG compression produces a structured stream that consists of two types

of data : *i*) coding parameters that provide the necessary information to decode the stream, for example the number of color components, quantization tables and Huffman tables, and *ii*) the entropy coded data. By hiding the coding parameters, the decoder will fail to decode the entropy coded data. In the JPEG stream, parameters are grouped into different types of data segments depending on what they specify. This includes the *Frame header*, *Scan header*, *Quantization table specification* and *Huffman table specification*. The number of variables in a data segment and their values vary with data segments and so the security provided by encrypting them varies in each case.

In this chapter we first review the structure of the JPEG stream and then examine the level of security that will be provided by encrypting different parts of the stream. We then identify the part that results in the highest level of security. As will be shown in Section 5.5, most parameters are easily predictable and so encrypting them does not provide high security. We show that encrypting the Huffman table specifications will provide high security with a very small computational overhead. An image viewer (`xv`) [13] used for the experiments recognized the encrypted file as the JPEG stream and produced the error message for the Huffman specifications.

5.2 JPEG Compression

JPEG compression supports different methods to compress image data [45]. The compression is either *lossy* or *lossless*. The lossy compression includes the *sequential DCT* and the *progressive DCT* modes, and lossless compression is achieved by the *Lossless* mode. In this chapter, we mainly consider the sequential DCT mode with Huffman coding, which is the most commonly used mode of operation.

JPEG compression consists of three stages : *i*) transform, *ii*) quantization and *iii*) entropy coding.

Transform

The image is divided into 8×8 blocks. Each block is transformed into real number coefficients using the Discrete Cosine Transform (DCT) [3].

Quantization

Next the DCT coefficients are quantized. This is done by dividing each coefficient by an integer value and then rounding the result.

Entropy coding

Finally the quantized coefficients are entropy coded. JPEG provides two different

types of coders, that is, Huffman coder and arithmetic coder. The commonly used coder is the Huffman coder. One of the reasons of common use of the Huffman coding in JPEG is that the JPEG arithmetic coding algorithm is patented.

Compression in the Sequential DCT mode

In the sequential DCT mode, 8×8 pixel blocks are sequentially scanned from left to right, and top to bottom of an image, and through the scan, each block is transformed, quantized and entropy coded independently.

Compressed images are normally stored as files. The JPEG files will have the format specified by the JPEG File Interchange Format (JFIF) [33].

5.2.1 Huffman Coding in JPEG

In the sequential DCT mode, to encode the quantized DCT coefficients of a block, first the DC coefficient is encoded and then the 63 AC coefficients are zig-zag scanned and encoded. The details of the encoding of the DCT coefficients are described in the following sections.

Encoding DC Coefficients

Encoding of the quantized DC coefficients is as follows.

	Algorithm 1 : Encoding DC coefficients
1 :	For all blocks (loop) :
2 :	Calculate the difference D_{diff} of the quantized DC coefficients D_{DC} in two consecutive blocks.
3 :	For D_{diff} obtain a category number C_{DC}, $0 \leq C_{DC} \leq 11$, using Table 5.1.
4 :	For D_{diff} obtain an index number T_{DC}, $0 \leq T_{DC} \leq 2^{C_{DC}} - 1$, using Table 5.1.
5 :	Huffman-encode C_{DC}.
6 :	Output T_{DC} as a C_{DC} bit binary value.

The method to obtain C_{DC} and T_{DC} is as follows.

- Choose C_{DC}, where range includes D_{diff}.

  The range is defined as follows.

  - If $D_{diff} < 0$, then $-(2^{C_{DC}} - 1) \leq D_{diff} \leq -(2^{C_{DC}-1})$
  - If $D_{diff} \geq 0$, then $(2^{C_{DC}} - 1) \geq D_{diff} \geq (2^{C_{DC}-1})$

- Choose T_{DC}, which indicates the position of D_{diff} in the range.

  For example, if $D_{diff} = -5$, then $C_{DC} = 3$ and $T_{DC} = 2$.

Table 5.1: Table of category numbers and index numbers.

Category number	D_{diff} for encoding DC, D_{AC} for encoding AC							
0	0							
1	-1	1						
2	-3	-2	2	3				
3	-7	-6	-5	-4	4	5	6	7
⋮	⋮							⋮
11	−2047	-2046	-2045	-2044	-2043	-2042	-2041	...
	0	1	2	3	4	5	6	...
	Index number (T_{DC} or T_{AC})							

Note that C_{DC} is the number of bits required for T_{DC}. For example, for $C_{DC} = 3$, the range of T_{DC} is $[0, 7]$ and three bits are used to represent the value.

Encoding AC Coefficients

The quantized AC coefficients D_{AC} in a block are zig-zag scanned, and encoded as follows.

	Algorithm 2 : Encoding AC coefficients
1 :	Initialize the run-length R to zero.
2 :	Zig-zag scan (loop) :
3 :	If $D_{AC} = 0$,
4 :	$R = R + 1$.
5 :	If D_{AC} is the last AC coefficient encoded in the block,
6 :	Huffman-encode *End Of Block* (EOB) code. (This code indicates that no more AC coefficients will be encoded in the block. If the encoded AC coefficient is not the sixty fourth coefficient, subsequent coefficients are not encoded.)
7 :	If $D_{AC} \neq 0$,
8 :	If $R \geq 15$,
9 :	Huffman-encode the code `0xF0`, that represents consecutive 15 zero coefficients, $\lfloor R/15 \rfloor$ times.
10:	$R = R \ mod \ 15$.
11:	Obtain category number C_{AC}, $0 \leq C_{AC} \leq 10$ using Table 5.1.
12:	Obtain index number T_{AC}, $0 \leq T_{AC} \leq 2^{C_{AC}} - 1$ using Table 5.1.
13:	Create an 8 bit value $a = 16R + C_{AC}$.
14:	Huffman-encode a.
15:	Output T_{AC} as a C_{AC} bit binary value.
16:	Initialize run-length R to zero.

Huffman Coding in JPEG

In the following, we review Huffman coding and describe the relationship between the Huffman code and the DCT coefficients.

Let $\mathcal{H}$ be the set of the Huffman codewords that is generated for an alphabet $\mathcal{A}$ with probability distribution $\mathcal{P}$. Let $\mathcal{M}$ be a function that maps the source symbol $a \in \mathcal{A}$ to the code word $h \in \mathcal{H}$, i.e. $\mathcal{M} : a \rightarrow h$, and let $\mathcal{M}^{-1}$ be the inverse of $\mathcal{M}$. Then the encoding and the decoding are shown as $h = \mathcal{M}(a)$, and $a = \mathcal{M}^{-1}(h)$, respectively.

The alphabet for the DC Huffman code is the set of category numbers C_{DC} which are used in the encoding. The alphabet only includes the category numbers corresponding to D_{diff} that appear in the image.

For the AC Huffman code, the alphabet is the set of eight bit values $16R + C_{AC}$,

corresponding to pairs of run-length and category number that appear when encoding the image.

When the encoder encodes an image, it constructs the Huffman code from the frequencies of the source symbols.

In the encoded bit stream, an index number of C_{DC} bits and C_{AC} bits follows a codeword corresponding to DC and AC coefficient, respectively. For the correct decoding, it is necessary to locate the beginning of a codeword and so the size of the index number following a codeword must be known. This size is determined by the source symbol that is encoded as a codeword right before the index number.

5.3 JPEG Stream

JPEG data is a structured stream where different parts of the stream are separated by markers. To maintain the conformance with the JPEG standard, the markers must be kept intact. The compressed image data consists of a *frame*, and the frame contains one or more *scan*. The structure of a JPEG stream is as follows.

Frame A *frame* begins with a *frame header* and contains one or more *scan* data.

The frame header may be preceded by one or more *table-specification* (optional) or miscellaneous *marker segments* (optional).

Scan A *scan* begins with a scan header and contains one or more entropy-coded data segments.

Each scan header may be preceded by one or more *table-specification* or miscellaneous *marker segments*.

The high-level structure of the compressed image data is shown in Table 5.2.

5.3.1 JPEG Data Components

There are several types of data segments which contain coding parameters.

1. A frame header contains the information of the entire image such as the image geometry and the number of color components.

2. A *quantization table specification* specifies the quantization values used for the 8×8 DCT coefficients. Different quantization tables can be used for the luminance and the chrominance components.

Table 5.2: The high-level structure of the JPEG stream.

[Table specifications]	
Frame header	
Scan 1	[Table specifications]
	Scan header 1
	Entropy coded segment 1
Scan 2	[Table specifications]
	Scan header 2
	Entropy coded segment 1
⋮	⋮
Scan last	[Table specifications]
	Scan header last
	Entropy coded segment last

3. A *Huffman table specification* provides the necessary parameters to construct a Huffman table used for the decoding. In JPEG, two types of Huffman tables are used : a *DC Huffman table* for DC coefficients and an *AC Huffman table* for AC coefficients. Different DC/AC Huffman table pairs are commonly used for the luminance and the chrominance components for color images.

4. A scan header, preceding an entropy coded data segment, specifies which quantization and Huffman tables to be used for the decoding and contains the structural information of the following entropy coded segment.

The above four data segments are essential components of a lossy JPEG data stream. In the following, the contents of the frame header, the scan header, the quantization table specification, and the Huffman table specifications are described. We use the representations used in the JPEG standard document [45].

Frame Header

The frame header consists of the following information.

Frame header length (Lf) The length of the header in bytes including the header length.

Sample precision (P) The precision in bits for the samples.

Number of lines (Y) The number of lines in the source image.

Number of samples per line (X) The number of columns in the source image.

Number of image components in frame (Nf) The number of components in the source image.

Image components Information about the image components. The number of image components is given by the *Number of image components in frame*. For the following four items composing *Image components*, $i \in \{1, 2, 3, ..., Nf\}$.

Component identifier (C_i) Unique number assigned to each component.

Horizontal sampling factor (H_i) The factor that specifies the relationship between the component horizontal dimension and the number of samples per line in the source image.

Vertical sampling factor (V_i) The factor that specifies the relationship between the component vertical dimension and number of lines in the source image.

Quantization table destination selector (Tq_i) The identifier that specifies which one among the four quantization tables should be used in the dequantization.

The values of the parameters in the frame header are shown in Table 5.3.

Table 5.3: Frame header.

parameter	bits	values
Lf	16	3 + 3 × Number of components
P	8	8 (baseline), 8,12 (extended,progressive), 2-16 (lossless)
Y	16	0-65,535
X	16	1-65,535
Nf	8	1-255 (baseline, extended,lossless), 1-4 (progressive)
C_i	8	0-255
H_i	4	1-4
V_i	4	1-4
Tq_i	8	0 (lossless), 0-3 (other processes)

Scan Header

The scan header consists of the following information.

Scan header length (Ls) The length of the header in bytes including the header length.

Number of image components (Ns) The number of source image components in the scan. For the following three items, $j \in \{1, 2, 3, ..., Ns\}$.

Scan component selector (Cs_j) The identifier that specifies the place of the components specified in the frame header in subsequent data segments.

DC entropy coding table destination selector (Td_j) The identifier that specifies one of the four possible DC entropy coding tables.

AC entropy coding table destination selector (Ta_j) The identifier that specifies one of the four possible AC entropy coding tables.

Start of spectral selection (Ss) The first DCT coefficient in each block in zig-zag order.

End of spectral selection (Se) The last DCT coefficient in each block in zig-zag order.

Successive approximation bit position high (Ah) In the sequential DCT mode, the value is always 0.

Successive approximation bit position low (Al) In the sequential DCT mode, the value is always 0.

The values of the parameters in the scan header are shown in Table 5.4.

Table 5.4: Scan header.

parameter	bits	values
Ls	16	6 + 2 × Number of image components
Ns	8	1-4
Cs_j	8	0-255
Td_j	4	0-1 (baseline), 0-3 (other)
Ta_j	4	0-1 (baseline), 0-3 (seq. extended/progressive), 0 (lossless)
Ss	8	0 (sequential), 0-63 (progressive), 1-7 (lossless)
Se	8	63 (sequential),Ss-63 (progressive), 0 (lossless)
Ah	4	0-13 (progressive), 0 (other)
Al	4	0 (sequential), 0-13 (progressive), 0-15 (lossless)

There are two types of data segments which define the parameters of the quantization and the entropy coding, that is, the quantization table and Huffman table specification segment. In the following, these two types of data segments are described.

Quantization Table Specification

This data segment includes the following information.

Length (Lq) The length of the quantization table specifications in bytes including the header length.

Quantization table element precision ($Pq[t]$) The precision of the quantization value which is either 0 (8 bits) or 1 (16 bits).

Quantization table id ($Tq[t]$) The identifier of the table.

Quantization table element ($Q_k[t]$) The 64 quantization values ($k \in \{1, 2, 3, ..., 64\}$).

The contents of the table are shown in Table 5.5.

Table 5.5: Quantization table specification (o is the number of quantization tables in the quantization table specifications).

parameter	bits	values
Lq	16	$2 + \sum_{t=1}^{o}(65 + 64 \times Pq[t])$
$Pq[t]$	4	0,1
$Tq[t]$	4	0-3
$Q_k[t]$	8,16	0-255, 0-65535

Huffman Table Specification

The Huffman table specification consists of the following parameters.

Length (Lh) The length of the Huffman table specifications in bytes including the header length. A Huffman table specification includes o table specifications (in the following part, $t \in \{1, 2, 3, ..., o\}$).

Table class ($Tc[t]$) DC table (0) or AC table (1).

Identifier ($Th[t]$) The identifier of the table.

Number of Huffman codewords of length i ($L_i[t]$) The number of Huffman codewords for each of the 16 possible lengths (i.e. $i \in \{1, 2, ..., 16\}$).

Value ($V_{i,j}[t]$) The value associated with each codeword of length i. If there are L_i Huffman codewords for length i, then there are L_i values for length i (i.e. $j \in \{1, 2, ..., L_i\}$, $m_t = \sum_{i=1}^{16} Number\ of\ Huffman\ codes\ of\ length\ i$).

Table 5.6: Huffman table specification.

parameter	bits	values
Lh	16	$2 + \sum_{t=1}^{o}(17 + \sum_{i=1}^{16} L_i[t])$
$Tc[t]$	4	0,1
$Th[t]$	4	0-3
$L_i[t]$	8×16	0-255
$V_{i,j}[t]$	$8 \times m_t$	0-255

The values of the parameters are shown in Table 5.6.

In the sequential DCT, $\mathcal{H}$, a set of codewords, is determined by the *Number of Huffman codes of length* i. The mapping between Huffman codewords and source symbols, $\mathcal{M}$, is determined by *Value* ($V_{i,j}$).

5.4 Encrypting Markers

There are markers that indicate the beginning of headers, table specifications and data segments. A possible way of making the stream unintelligible to the decoder is to encrypt the markers and give the key and their positions as part of the secret key information. However, *i*) the JPEG stream with encrypted markers will not be recognized as a JPEG stream, *ii*) the positions of the markers in the JPEG stream need to be given as the secret information and so this increases the size of the secret data, and *iii*) encrypting only markers will not entirely hide the data structure because the contents of some data segments such as quantization table specifications could be easily recognized in the JPEG stream even if the markers are hidden. Because of these reasons, encryption of markers is not considered.

5.5 Encryption of JPEG Components

Selective encryption reduces the computational cost by encrypting small amount of data. The information to be encrypted should be carefully chosen to minimize the amount of data to be encrypted while providing high security. The encrypted data should satisfy the following conditions.

Condition 1 Without the encrypted data, it is difficult to decode the JPEG stream.

Condition 2 It is difficult to derive the encrypted data from other information in the same JPEG stream.

Condition 3 The encrypted data is highly dependent on the image and so the corresponding data from similar images are not useful.

Condition 4 The search space of the encrypted data must be large.

Some of the coding parameters in the JPEG stream have small number of possible values. For example, the *Number of image components* in the frame header can be one (gray scale) or three (color image) and the *Table class* in the Huffman table specifications is either for DC or AC. The parameters such as the header length in the frame header is likely to be either 12 (gray) or 17 (color). If a parameter has a small number of possible values, it is easy to guess and verify the guess using the encrypted data. Hiding such parameters does not provide large search space and so it will have minimal effect on security.

Parameters that have relatively large number of choices are as follows.

- Image geometry
- Quantization table
- Huffman table specifications

In the following sections, we examine these parameters and show whether or not they satisfy the above four conditions.

5.5.1 Encrypting Headers

In this section, we examine the complexity of finding parameters in the encrypted headers. The range of the parameters in JPEG varies with the type of compression. In the following analysis, we consider the sequential DCT mode which is the most commonly used mode and assume that the markers are not encrypted.

Frame Header

The following shows the cost of finding the parameters in the encrypted frame header. We assume that all other data segments are unencrypted. The number of quantization and Huffman table specifications can be obtained from their markers. It is known that for sequential DCT coding, each luminance and chrominance component has one quantization table and two Huffman tables for DC and AC.

Frame header length (Lf) The length is known from the positions of the frame header marker and the marker of the next data segment.

Sample precision (P) It is either 8 or 12.

Number of lines (Y) The value can be between 0 and 65535 but this can be found if all DCT blocks are correctly decoded. In the sequential DCT mode, rows of blocks are encoded from the top to the bottom of the image and so if the sequence of decoded blocks is correctly partitioned, the image can be reconstructed by arranging the partitions from top to bottom. The partition can be easily found because consecutive decoded blocks in a partition should *look natural*. If two blocks belong to different partitions, the edge part between them would not match. If the number of blocks in horizontal direction is found, the geometry of the image will be known.

Number of samples per line (X) The value can be between 1 and 65535. However this can be found similar to above.

Number of image components in frame (Nf) The value can be between 1 and 255.

Component identifier (C_i) This can be calculated from the scan header lengths.

Horizontal sampling factor (H_i) The value can be between 1 and 4.

Vertical sampling factor (V_i) The value can be between 1 and 4.

Quantization table destination selector (Tq_i) The value can be between 0 and 3.

From above, the total number of possibilities for the combination of Nf,P,H_i,V_i and Tq_i is $255 \times 2 \times 4 \times 4 \times 4 \approx 2^{16}$.

The image geometry (*Number of lines* and *Number of samples per line* in the frame header) can vary with images and hiding this information provides a $65536 \times 65535 \approx 2^{31}$ search space. However, if all the 8×8 blocks are correctly decoded, the image can be easily reconstructed even if the width and the height of the image are encrypted. This violates **Condition 2**.

Scan Header

The cost of finding the JPEG parameters for the scan header is shown below.

Scan header length (Ls) The length is known from the positions of the frame header marker and the marker of the next data segment.

Number of image components (Ns) This can be calculated from Ls.

Scan component selector (Cs_j) The value is between 0 and 255.

DC entropy coding table destination selector (Td_j) The value is between 0 and 3.

AC entropy coding table destination selector (Ta_j) The value is between 0 and 3.

Start of spectral selection (Ss) The value is 0.

End of spectral selection (Se) The value is 63.

Successive approximation bit position high (Ah) The value is 0.

Successive approximation bit position low (Al) The value is 0.

From the above, the total number of possibilities for Cs_j, Td_j and Ta_j is $256 \times 4 \times 4 = 2^{12}$. We note that typical values of Nf for a color image is three and so Cs_j will be 0,1 and 2 although Cs_j has the range $[0, 255]$. Hence, the number of possibilities for this parameter is small and **Condition 4** is not satisfied.

5.5.2 Encrypting Quantization Table Specifications

Quantization tables vary with images and values in the table depend on the image and the compression quality. However even if the table entries are not correct, it is possible to recover a reasonable quality image. The experimental results are shown in Section 5.7.4. The quantization table can be closely approximated by the example quantization table given in the JPEG specification and so **Condition 3** is not satisfied for this table.

For successful de-quantization, the correct Huffman decoding, i.e. correct Huffman table, is required.

5.5.3 Encrypting Huffman Table Specifications

If Huffman tables are hidden, the decoding of the data segments will fail. The Huffman codes are constructed from the Huffman table specifications in the JPEG data using the algorithm given in the JPEG standard document [45]. The examples in Section 5.8.1 show that parameters *Number of Huffman codes of length i* and *Values associated with*

each Huffman code vary not only with images but also with the compression quality and are critical for correct re-construction of the Huffman table in the decoder. They are also sensitive to the bit change of the table entries as shown in the experiment results in Section 5.7.6 and 5.7.5.

One advantage of encrypting the Huffman table specifications is that the size of encrypted data, compared with the size of the JPEG file, is very small. The sizes of the JPEG files and the Huffman table specifications in the above examples are shown in Table 5.7.

Table 5.7: Examples of sizes of encrypted Huffman table specifications.

The JPEG file size	Huffman table specifications
39246 bytes (Quality=75%)	161 bytes (0.4 %)
23027 bytes (Quality=50%)	148 bytes (0.65 %)

The above analysis shows that only Huffman table specifications satisfy Condition 1 to 4. For other data segments, some of the parameters can be derived from other parts of data in the same stream or from publicly available data, such as examples in the JPEG standard, and others only provide small search space.

5.6 Security of Huffman Code

In this section, we examine security of encrypting Huffman code in more details. The source alphabet for the DC Huffman code is the category numbers C_{DC} and for the AC Huffman code is the 8 bit values calculated from R and C_{AC}. The codewords are determined using the probability distribution of these values. For correct decoding of the JPEG stream, *i*) codewords of the Huffman code and *ii*) correct mapping between the codewords and source symbols must be known. When table specifications are encrypted, this information is hidden. We assume that the entire Huffman table specifications are hidden and estimate how much information can be obtained under the following attacks.

1. The attacker does not know the image that is encrypted and tries to find the hidden parameters by exhaustive search.

2. The attacker has some knowledge about the image that is encrypted and has access to images that are similar to the encrypted one.

First we examine the first attack and estimate the cost of the exhaustive search and then, consider the second case and examine the chance of success. Finally we discuss

applicability of attacks similar to those against arithmetic coding encryption schemes [11, 17, 40, 56, 113, 112] to Huffman code encryption system proposed here.

5.6.1 Complexity of Recovering the Huffman Table Using Exhaustive Search

Let n be the number of source symbols. Then to recover the Huffman table, the following information is required.

- The number of Huffman codewords L_i of each length $i \in \{1, 2, ..., 16\}$. We have $\sum_{i=1}^{16} L_i = n$.
- The set $\mathcal{A}$ of source symbols. For DC, possible source symbols $a \in \mathcal{A}$ are $0 \leq a \leq 11$. For AC, possible source symbols are $a = 16R + C_{AC}$ where $0 \leq R \leq 15$ and $0 \leq C_{AC} \leq 10$. In the Huffman code of an image, $\mathcal{A}$ only includes the source symbols which have appeared in the encoding.

  The number of the source symbols n can be derived from the size of the Huffman table definition and so n is public. In the DC case, $1 \leq n \leq 12$ and for the AC case, $1 \leq n \leq 176$.
- The mapping between the Huffman code and the source alphabet.

The recovery can be divided into the following problems.

Problem 1 Finding the number of codewords L_i for each length $i \in \{1, 2, ..., 16\}$ where $\sum_{i=1}^{16} L_i = n$ is known.

Problem 2 Finding n, the number of source symbols, where $1 \leq n \leq 12$ and $1 \leq n \leq 176$ for DC and AC, respectively.

The cost is $\binom{12}{n}$ for DC and $\binom{176}{n}$ for AC.

Problem 3 Finding the mapping between n source symbols and codewords requires at most $n!$ tries.

The Huffman code is obtained by constructing a Huffman binary tree. In a Huffman binary tree for n source symbols, there are n leaf nodes, each corresponding to a source symbol. Each node is assigned 0 or 1 and a codeword corresponds to the path from the root to a leaf. It is known that for a given n, there are $\binom{2n-2}{n-1}/n$ possible binary

trees with n leaves [49]. If the probability distribution of source symbols is not known, the attacker needs to try all possible trees to find the codewords and so the cost of the attack is the cost of exhaustive search over all possible Huffman codes. Once the correct binary tree is found, then the number of codewords for length $i \in \{1, 2, 3, ..., 16\}$ can be obtained.

The complexity of recovering the DC and AC tables are shown in the following sections.

Recovering the DC Table

To solve *Problem 1*, the maximum value of n is 12 and so the cost is at most $\binom{24-2}{12-1}/12 = \binom{22}{11}/12 = 705432 \approx 2^{19}$.

The number of symbols n depends on the image and the compression quality level and becomes smaller as the quality level drops. The reason for this can be seen from the distribution of differential DC value over category ranges given in Section 5.8. As the quality level drops, the differential DC values moves toward zero and so only the smaller category numbers are required to encode the values. The range of n is not very large and the cost of solving *Problem 2* is maximized when $n = 6$ and the maximum cost is given by $\binom{12}{6} = 924 \approx 2^{10}$.

For *Problem 3*, the maximum cost is when $n = 12$, and the number of possible assignments of the source symbols is $12! \approx 2^{29}$.

Recovering the AC Table

Recovering the AC table is similar to the DC case. To solve *Problem 1*, the maximum value of n is 176 and so the cost is smaller than $\binom{352-2}{176-1}/176 = \binom{350}{175}/176$.

The cost of solving *Problem 2* is maximized when $n = 88$ and the cost is $\binom{176}{88} \approx 2^{172}$. To solve *Problem 3*, the maximum cost of finding the mapping between the Huffman code and the source alphabet is $n! = 176!$.

To find L_i, it is necessary to find the source symbol distribution. This is more difficult in AC case than the DC case because the source symbols in the AC case are pairs of run-length and category number and so compared to the DC case there is one more unknown variable.

The Total Complexity of Recovering the Huffman Table

From above analysis, we note that both DC and AC tables must be given to the decoder and so the total cost is the product of the cost of the cases.

For **Problem 1**, **Problem 2** and **Problem 3**, the maximum costs are approximately $2^{19}\binom{350}{175}/176$, 2^{182} and $2^{29}176!$, respectively.

As can be seen from example data in Sections 5.8 and 5.8.1, there are some similarities between the distributions of differential DC values for the same image compressed with different quality levels. This means that the Huffman table specifications will not be completely independent and so if an attacker has an encrypted image of lower quality and a valid secret key, finding the Huffman table for the same image with higher quality would cost less than the maximum given above.

5.6.2 Security Analysis : Using the Information from Similar Images

In the following, we assume that the attacker has some knowledge about the encrypted image. Assuming the same size source alphabet is used, if a different Huffman code is used for the encoding and decoding, the original message and the decoded one will not be the same. The decoded symbols will be different from the encoded ones and the number of symbols in the original and decoded messages will be also likely to differ.

We consider the following case. There are two similar images. For one of them, all the information required for decoding, such as Huffman and quantization table specifications, is known but the corresponding information for the other image is encrypted. The question is whether or not the attacker can correctly decode the second image.

Suppose there are two alphabets $\mathcal{A}_1$ and $\mathcal{A}_2$ of the same size with the probability distributions $\mathcal{P}_1$ and $\mathcal{P}_2$. Then for the corresponding Huffman codes $\mathcal{H}_{\mathcal{P}_1}$ and $\mathcal{H}_{\mathcal{P}_2}$, with the mapping functions $\mathcal{M}_1 : a_1 \rightarrow h_1$ and $\mathcal{M}_2 : a_2 \rightarrow h_2$, where $a_1 \in \mathcal{A}_1$, $h_1 \in \mathcal{H}_{\mathcal{P}_1}$, $a_2 \in \mathcal{A}_2$, and $h_2 \in \mathcal{H}_{\mathcal{P}_2}$, we consider the following cases.

1.	$\mathcal{A}_1 \neq \mathcal{A}_2$	In this case the two alphabet sets are different.
2.	$\mathcal{A}_1 = \mathcal{A}_2$, $\mathcal{P}_1 \neq \mathcal{P}_2$	The two alphabet sets are the same but the set of probabilities are different.
3.	$\mathcal{A}_1 = \mathcal{A}_2$, $\mathcal{P}_1 = \mathcal{P}_2$, $\mathcal{M}_1 \neq \mathcal{M}_2$	The two alphabet sets and the set of probabilities are the same but they are allocated to the alphabet with a different mapping.
4.	$\mathcal{A}_1 = \mathcal{A}_2$, $\mathcal{P}_1 = \mathcal{P}_2$, $\mathcal{M}_1 = \mathcal{M}_2$	The two alphabet sets are the same. The two probability distributions and the mappings are the same too.

If $\mathcal{A}_1 = \mathcal{A}_2$, $\mathcal{P}_1 = \mathcal{P}_2$ and $\mathcal{M}_1 = \mathcal{M}_2$, then the decoder can correctly decode the encoded data.

Same Codewords and Different Mapping

This corresponds to the case 3 above. In the JPEG stream, Huffman code word is followed by the encoded index number. If $\mathcal{M}_1 \neq \mathcal{M}_2$, that means the decoded category number is wrong. Since the category number determines the number of bits required to represent the subsequent index number, the wrong category number means the incorrect decoding of the index number. As a result, the decoded index number is incorrect and the decoder fails to correctly position the next codeword and resulting in the loss of synchronization. In addition, since the source symbols include the run-length of the AC coefficients, if the run-length of zero coefficients is wrong, the decoded non-zero coefficients will be allocated in the wrong position in the 8×8 DCT coefficient block.

If the two mappings are partially equal, as long as the encoder and the decoder use the same mapping, the decoding will be correct. However, once a part which uses a different mapping starts, the Huffman decoder will lose synchronization and the rest of decoding will fail.

Different Codewords

Use of different codewords will result in the failure of the decoding. If the codewords are partially the same, as long as the encoded streams contain codewords and mapping which the encoder and the decoder agree, the decoder will correctly decode. If the

decoder reaches a codeword which the encoder has used a different mapping on, the synchronization will be lost.

Images and Huffman Codes

The relationship of two Huffman codes depends on the following parameters.

The size of $\mathcal{A}$ The size of $\mathcal{A}_1$ and $\mathcal{A}_2$ determines the number of codewords, and so different sizes mean different number of codewords.

This number can be obtained from the size of the Huffman table specification segment and so this parameter is known.

The source symbols in $\mathcal{A}$ In JPEG, the source symbols are determined by the set of *i*) distinct category numbers for DC and *ii*) distinct pairs of category number and run-length for AC.

The probability distribution $\mathcal{P}$ If $\mathcal{P}_1$ and $\mathcal{P}_2$ consist of the same probabilities while they are assigned to different source symbols, then $\mathcal{H}_1 = \mathcal{H}_2$ but $\mathcal{M}_1 \neq \mathcal{M}_2$.

This is determined by the frequencies of *i*) distinct category numbers for DC and *ii*) distinct pairs of category number and run-length for AC.

The mapping $\mathcal{M}$ This is also determined by *i*) frequencies of distinct category numbers for DC and *ii*) frequencies of distinct pairs of category number and run-length for AC.

If there is a difference in the edges of the two images, then AC values will be different and also the DC values may differ. This will result in different probability distributions for coefficients and different run-lengths and so the different Huffman codes for the two images. As shown in Section 5.7.1, the AC Huffman table is sensitive to changes in the high frequency components. Since smoothing changes the high frequency components over the whole image, the category numbers of the AC coefficients and the run-lengths of zero coefficients will change. As a result, the probability distribution of the source symbols of the AC Huffman code changes. This implies that if the difference in two images is local and the number of blocks that are different is small, the probability distribution of source symbols will not change very much even if the difference is relatively large. Hence in this case the Huffman table will not change.

For example, consider a service which provides maps. The maps are regularly updated and the new maps are delivered to the customers in encrypted form. Assuming

that the attacker owns an old map and then he intercepts a new encrypted map sent to someone else. If the difference between the old and the new map is small, it is feasible for him to decode the new map using the Huffman table specifications of the old one.

As shown in Section 5.7.2, Huffman code is sensitive to the quality level because the Huffman code is derived from the probability distribution of category numbers and the distribution of category numbers is determined by the quantized coefficients. The categories correspond to the partitions over an interval $[-2047, 2047]$ and quantized coefficients are in the interval. The range of each partition is constant regardless of image quality although values of quantized coefficients vary with image quality. This means that different image quality will result in different number of coefficients in each partition (category). The number of coefficients in a partition determines the probability of the corresponding category number and so different image quality will result in different probability distribution of category numbers.

It is shown by Fraenkel and Klein [28] that given the Huffman encoded stream of a natural language text the problem of decoding the stream, that is, finding the codewords is NP-complete. It has also been shown that if a bit stream is a sequence of pairs of a codeword of a prefix code and a short random bit string then the decoding problem is NP-complete. This is very similar to the entropy coded stream in JPEG where the random bit strings correspond to index numbers. However, in the case of JPEG, the bit strings are not random. The frequency (i.e. probability) of index numbers corresponds to the number of DCT coefficients in the two intervals which have the same size in the negative and the positive parts as shown in Table 5.1. It is known that the distribution of DCT coefficients is the generalized Gaussian [25, 47, 54], the density function of which is given by [26]

$$p(x) = \left[\frac{\nu\eta(\nu,\sigma)}{2\Gamma(1/\nu)}\right] exp(-[\eta(\nu,\sigma)|x|]^{\nu})$$

where

$$\eta(\nu,\sigma) = \sigma^{-1}\left[\frac{\Gamma(3/\nu)}{\Gamma(1/\nu)}\right]^{1/2}$$

and so the distribution of index numbers in the interval is close to Gaussian. Hence the index numbers close to 0, both the negative and the positive parts, have higher frequencies. The distribution of index numbers in four Huffman codes (to encode DC and AC coefficients of luminance and chrominance components) of `lena.pgm` is shown in Figure 5.1. Since the distribution of index numbers is skewed and not uniform, the analysis shown in [28] does not apply to JPEG and the cost of finding the Huffman code will be smaller than the analysis in [28].

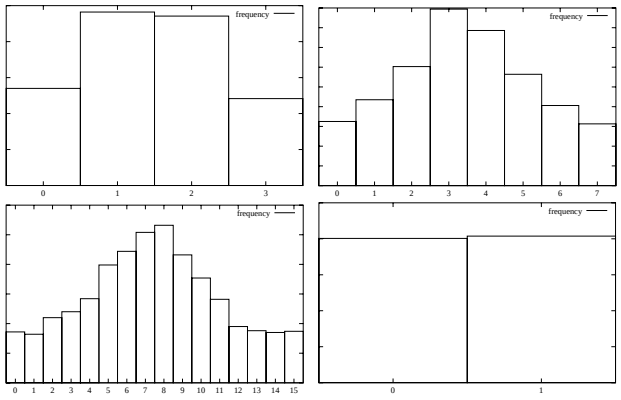


Figure 5.1: Distribution of index numbers for four Huffman codes.

5.6.3 Huffman Coding and Arithmetic Coding

There are encryption schemes for adaptive arithmetic coding systems in which the probability distribution of source symbols is a secret [120, 62]. For these schemes, there are known attacks [10, 11] that try to synchronize the adaptive model by sending a chosen bit string. The Huffman coder in JPEG uses a non-adaptive model and so similar attacks are not applicable.

5.6.4 Chosen plaintext and ciphertext attacks

By choosing input images, the attacker can generate the Huffman specification of his choice and mount a chosen plaintext attack to find the key. Similarly, he can mount a ciphertext attack by choosing ciphertexts (encrypted images). The Huffman specification is encrypted by an encryption algorithm such as AES and so the cost to find the key largely depends on the algorithm which is used for encryption.

5.7 Experiments

We conducted various experiments to verify the security analysis. Section 5.7.1 and 5.7.2 show the sensitivity of Huffman tables to smoothing and image quality levels. In Section 5.7.3 the probability distributions of binary symbols in the Huffman coded stream are shown. The stream consists of codewords of Huffman code and index numbers. Although index numbers will not have uniform distribution, the distribution of binary symbols in the stream can be considered as uniform. Section 5.7.4 and 5.7.5 show the decoding experiments of JPEG streams which have modified quantization and Huffman table specifications, respectively. Finally the decoding result of the JPEG

stream using encrypted Huffman table specifications is shown in Section 5.7.6.

5.7.1 Tables with Different Smoothness

This experiment used the *smoothing* function of *cjpeg* to modify the original image. The comparison was done between the 75% quality level image and the smoothed image with the same quality. The procedure of the experiment was as follows.

1. Create the JPEG file `lena_75.jpg` using `cjpeg` command from `lena.ppm`.

   ```
   cjpeg -optimize -baseline -quality 75 <lena.ppm >lena_75.jpg
   ```

2. Create the JPEG file `lena_75_1.jpg` using `cjpeg` command from `lena.ppm`.

   ```
   cjpeg -optimize -baseline -quality 75 -smooth 1 <lena.ppm >lena_75_1.jpg
   ```

   The smooth value 0 means no smoothing. (The algorithm is not described by the `cjpeg` manual.)

3. Compare the corresponding Huffman specification segments of the two files, i.e. DC table for luminance, AC table for luminance, DC table for chrominance, and AC table for chrominance,

The PSNR of both files was 50.23 dB. The quantization tables of the two images were the same. The DC tables of luminance and chrominance and AC table of luminance were the same. The AC table for chrominance were as follows.

Without smoothing,

```
00000000 : 00 2b 11 00 02 02 02 02 02 02 02 02 02 03 00 03
00000010 : 00 00 00 00 01 02 11 03 21 12 31 04 41 22 32 51
00000020 : 61 13 42 23 71 33 81 91 52 a1 f0
```

and $Lh = 43$,$Tc = 1$,$Th = 1$, $L_i = 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 0, 3, 0, 0, 0$, $i = 1, 2, ..., 16$, and the values of $V_{i,j}$ are as follows:

		$j=$		
		1	2	3
	2	0	1	
	3	2	17	
	4	3	33	
	5	18	49	
	6	4	65	
$i=$	7	34	50	
	8	81	97	
	9	19	66	
	10	35	113	
	11	51	129	145
	13	82	161	240

With smoothing,

```
00000000 : 00 2b 11 00 02 02 02 02 02 02 01 05 00 03 00 03
00000010 : 00 00 00 00 01 02 11 03 21 12 31 04 41 22 32 51
00000020 : 13 23 42 61 71 33 81 91 52 a1 f0
```

and $Lh = 43$,$Tc = 1$,$Th = 1$, $L_i = 0, 2, 2, 2, 2, 2, 2, 1, 5, 0, 3, 0, 3, 0, 0, 0$, $i = 1, 2, ..., 16$, and the values of $V_{i,j}$ are as follows:

		$j=$				
		1	2	3	4	5
	2	0	1			
	3	2	17			
	4	3	33			
	5	18	49			
	6	4	65			
$i=$	7	34	50			
	8	81				
	9	19	35	66	97	113
	11	51	129	145		
	13	82	161	240		

If Huffman tables of `lena` without smoothing replaces the ones with smoothing, the viewer `xv` produces the error *Corrupt JPEG data: bad Huffman code.* The produced

image is shown in Figure 5.2. This shows that with small difference in encoding the same image with or without smoothing, one Huffman table cannot be used for the other.

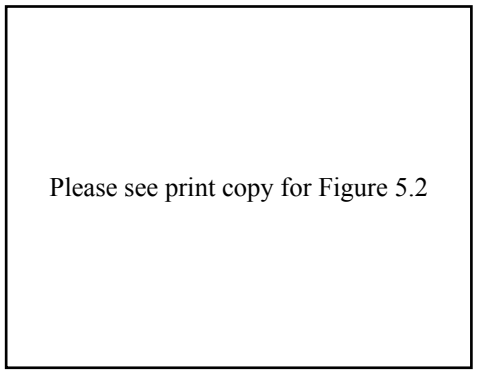


Figure 5.2: The image with the Huffman AC chrominance table of the image with smoothing.

5.7.2 Tables with Different Quality Levels

This experiment shows sensitivity of the Huffman code to image quality, i.e. the quantization divisors. The image used was `lena.ppm`. The experiment was as follows.

1. Create the JPEG file `lena_75.jpg` using `cjpeg` command from `lena.ppm`.

   ```
   cjpeg -optimize -baseline -quality 75 <lena.ppm >lena_75.jpg
   ```

2. Create the JPEG file `lena_74.jpg` using `cjpeg` command from `lena.ppm`.

   ```
   cjpeg -optimize -baseline -quality 74 <lena.ppm >lena_74.jpg
   ```

3. Compare the corresponding Huffman specification segments of these two files, i.e. DC table of luminance, AC table of luminance, DC table of chrominance, and AC table of chrominance,

The PSNR of the two files were 47.26 dB. The DC tables for both luminance and chrominance were the same. The AC tables for luminance and chrominance were as follows.

Luminance : Quality 75%

```
00000000 : 00 40 10 00 01 03 03 03 02 04 03 07 03 03 04 03
00000010 : 00 01 05 01 02 03 11 00 04 21 05 12 31 41 51 06
00000020 : 13 22 61 32 71 81 14 23 91 a1 b1 c1 f0 42 d1 e1
00000030 : 07 15 52 24 33 62 f1 34 43 82 72 16 26 92 a2 b2
```

Luminance : Quality 74%

```
00000000 : 00 3f 10 00 01 03 03 03 02 04 04 04 05 03 04 03
00000010 : 00 02 03 01 02 03 11 00 04 21 05 12 31 41 51 06
00000020 : 13 22 61 32 71 81 91 14 a1 b1 c1 23 42 d1 e1 f0
00000030 : 07 15 52 24 33 62 f1 34 43 82 16 72 26 92 b2
```

Chrominance : Quality 75%

```
00000000 : 00 2b 11 00 02 02 02 02 02 02 02 02 02 03 00 03
00000010 : 00 00 00 00 01 02 11 03 21 12 31 04 41 22 32 51
00000020 : 61 13 42 23 71 33 81 91 52 a1 f0
```

Chrominance : Quality 74%

```
00000000 : 00 2b 11 00 02 02 02 02 02 02 01 04 02 02 03 00
00000010 : 00 00 00 00 01 02 11 03 21 12 31 04 41 22 32 51
00000020 : 13 23 42 61 71 81 33 91 52 a1 f0
```

If the Huffman tables of 75% quality replaces the ones of a 74% quality JPEG file, the viewer `xv` produces the error *Corrupt JPEG data: bad Huffman code*. The resulting image is shown in Figure 5.3. This shows that if different JPEG quality levels are used, the two compressed images obtained from the same image produce different Huffman tables and one Huffman table cannot replace the other.

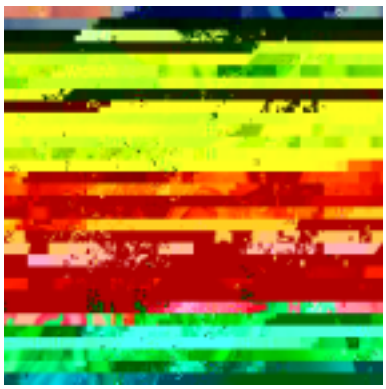


Figure 5.3: 74% quality image with 75% quality Huffman AC tables.

5.7.3 Probability Distribution of Binary Symbols

The following graphs show the probability distribution of n bit binary symbols. Figure 5.4 shows the probability distribution of one bit (left) and two bit (right) binary

symbols for the entropy coded data segment. Figure 5.5 and 5.6 show the probability distribution of three and four bit, and five and six bit binary symbols for the entropy coded data segment, respectively. The variances of probabilities for n bit symbols are shown in Table 5.8. An example of the bit sequence of the entropy coded data segment is shown below. From Figure 5.4, 5.5 and 5.6, it can be seen that the distribution of binary symbols are close to uniform and so it will be resistant against statistical analysis.

```
11010000 10010010 00101001 10101011 01010100 01001111 01001100 00011010
01101001 00011000 10000000 01111101 11101011 11000100 11001101 01111001
00101110 01000111 11010101 10100100 00100010 10010101 10000001 11011010
10010011 01110100 00101011 00100110 00101001 01001101 00110100 00000011
...
```

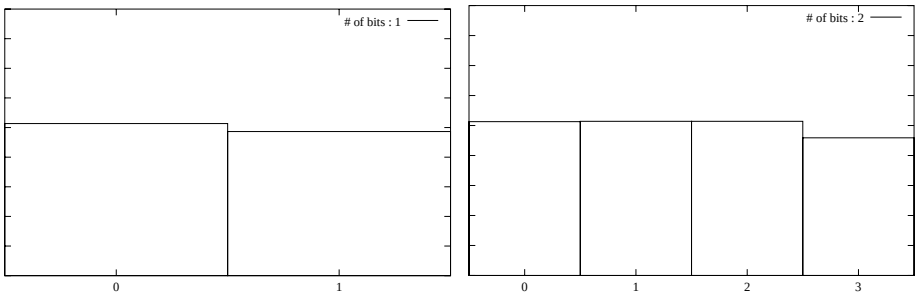


Figure 5.4: Probability distribution of one bit binary symbols (left) and two bit binary symbols (right).

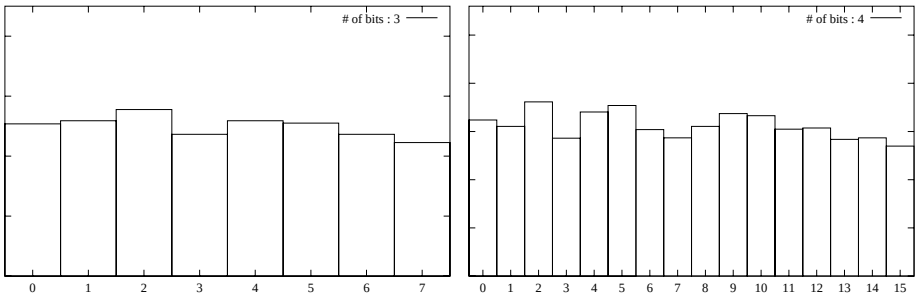


Figure 5.5: Probability distribution of three bit binary symbols (left) and four bit binary symbols (right).

5.7.4 Modification of Quantization Table Specifications

The following shows an example of the change of the quantization table specifications.

The original quantization table specifications are as follows.

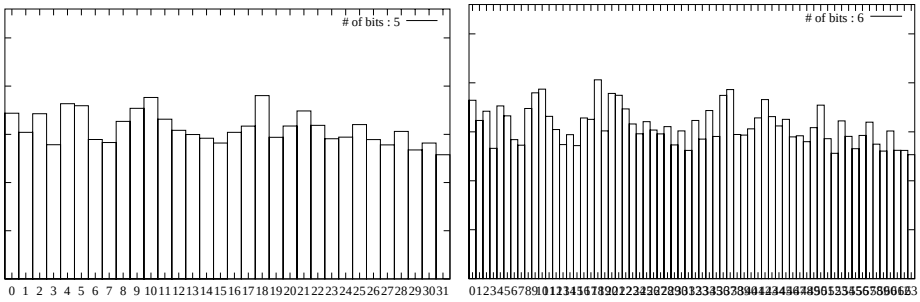


Figure 5.6: Probability distribution of five bit binary symbols (left) and six bit binary symbols (right).

Table 5.8: Variances of probabilities for n bit symbols.

bits	variance
1	0.16
2	0.034
3	0.0082
4	0.0020
5	0.00049
6	0.00012
7	0.000031
8	0.000008

```
DQT
len 67 Pq 0 Tq 0
   8   6   6   7   6   5   8   7   7   7   9   9   8  10  12  20
  13  12  11  11  12  25  18  19  15  20  29  26  31  30  29  26
  28  28  32  36  46  39  32  34  44  35  28  28  40  55  41  44
  48  49  52  52  52  31  39  57  61  56  50  60  46  51  52  50
DQT
len 67 Pq 0 Tq 1
   9   9   9  12  11  12  24  13  13  24  50  33  28  33  50  50
  50  50  50  50  50  50  50  50  50  50  50  50  50  50  50  50
  50  50  50  50  50  50  50  50  50  50  50  50  50  50  50  50
  50  50  50  50  50  50  50  50  50  50  50  50  50  50  50  50
```

The modified quantization table specifications are as follows.

```
DQT
len 67 Pq 0 Tq 0
  16  16  16  16  16  16  16  16  16  16  16  16  16  16  16  16
  16  16  16  16  16  16  16  16  16  16  16  16  16  16  16  16
```

```
  16  16  16  16  16  16  16  16  16  16  16  16  16  16  16  16
  16  16  16  16  16  16  16  16  16  16  16  16  16  16  16  16
DQT
len 67 Pq 0 Tq 1
  16  16  16  16  16  16  16  16  16  16  16  16  16  16  16  16
  16  16  16  16  16  16  16  16  16  16  16  16  16  16  16  16
  16  16  16  16  16  16  16  16  16  16  16  16  16  16  16  16
  16  16  16  16  16  16  16  16  16  16  16  16  16  16  16  16
```

The resulting images are shown in Figure 5.7. The PSNR of the modified image (right) is 17.6 dB with respect to the original image (left). Although the PSNR is low, the contents of the image are intact and so hiding quantization tables is not effective.

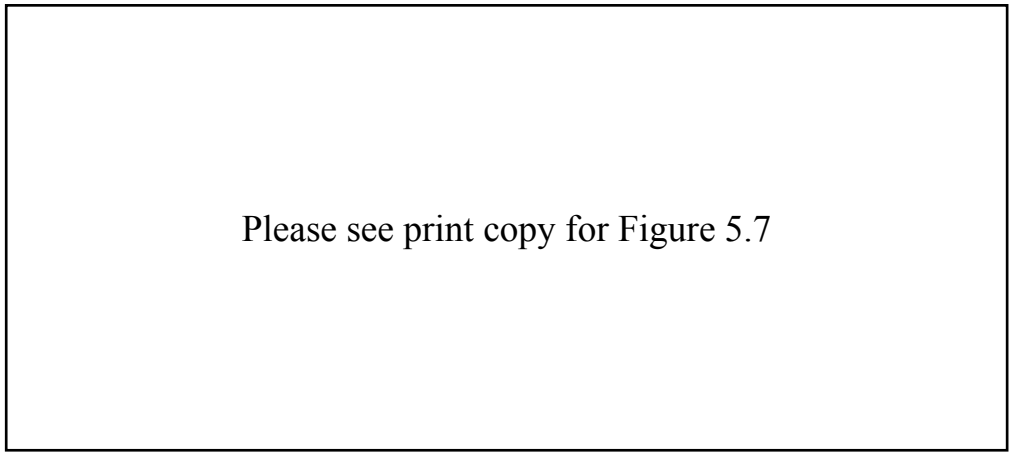


Figure 5.7: *Decoding with different quantization tables: the original image (left) and recovered image using different quantization tables (right).*

5.7.5 Modification of Huffman Table Specifications

The following experiment result shows an example of the change of the Huffman table specifications.

The first Huffman table specification is as follows.

```
DHT
len 26
 Tc 0 Th 0 L    1   0   2   3   1   0   0   0   0   0   0   0   0   0   0   0
  1:   0
  3:   4   5
```

```
4:   2   3   6
5:   1
```

and the last line was changed to

```
5:   3
```

In Figure 5.8, the original image is on the left. The image on the right has a modified Huffman table and results in `xv` producing the error message "corrupted Huffman table". Modifications may result in the complete failure of the decoding, i.e. no output image. The Huffman table specification is very sensitive to change and the decoding fails even if the change is small.

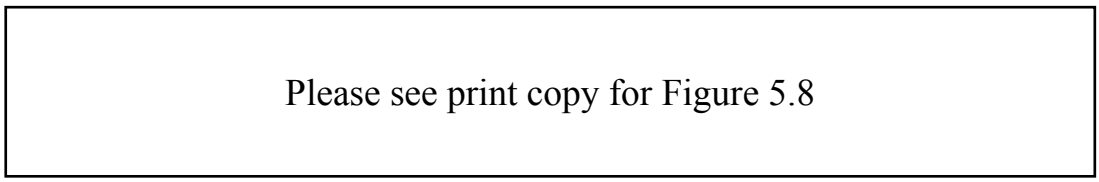


Figure 5.8: *Destruction of Huffman table: Viewing the original image (left) and the image with "corrupted" Huffman table (right) using* `xv`.

5.7.6 Encryption of Huffman Table Specification

The Huffman table specifications in a JPEG file are encrypted using DES 8-bit CFB mode [129, 38]. DES can be replaced by AES. Their sizes of these specifications are multiples of 8 bits (i.e. bytes), and so any encryption algorithm which can encrypt arbitrary number of bytes can be used. If we attempt to display the file using the command `xv`, it results in the error message "Bogus Huffman table definition" and fails to display the file.

5.8 Distribution of Differential DC Values

The DC Huffman table is determined by the distribution of differential DC values. If each image has the distinct distribution and the difference among images is large, then the resulting Huffman table differs largely and the difficulty of recovering the encrypted table from known tables will increase.

In the following experiments, we consider two images: lena and pepper. We compare the distribution of differential DC values of these images with various compression quality. The distributions of differential DC values (without quantization) are shown

in Figure 5.9. The x-axis is the differential DC value and the y-axis is the frequency of each value.

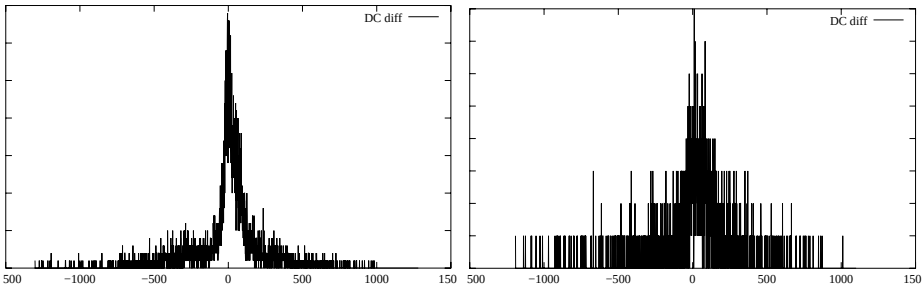


Figure 5.9: *Distributions of differential DC values of* `lena.pgm` *(left) and* `pepper.pgm` *(right).*

The distributions of differential DC values (with quantization) over intervals specified by the category numbers are shown in Figure 5.10, 5.11, 5.12, 5.13, and 5.14.

The x-axis is the category and the y-axis is the frequency of the differential DC values which are in the range of the category number.

If the quantization value is 2, the appeared symbols are $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. For two symbols X_1 and X_2 with their length L_1 and L_2, respectively, if $L_1 > L_2$, the probabilities of X_1 and X_2 will satisfy $P(X_1) \leq P(X_2)$.

In the `lena` case, the probabilities are $P(5) \geq \{P(3), P(4), P(6), P(7), P(8)\} \geq P(9) \geq P(2) \geq P(1) \geq P(0) \geq P(10)$.

The two graphs showed different distributions although they both had zero mean and similar ranges, and so they resulted in two different Huffman codes. From the graphs, it follows that distribution can be roughly known from other similar images.

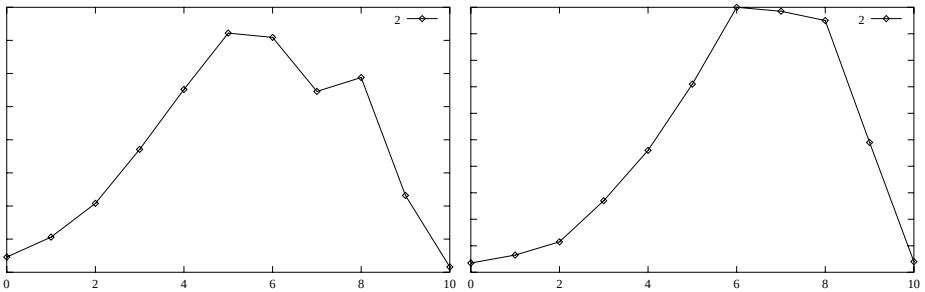


Figure 5.10: *Distributions of differential DC values of* `lena.pgm` *(left) and* `pepper.pgm` *(right) for* Q_1*=2.*

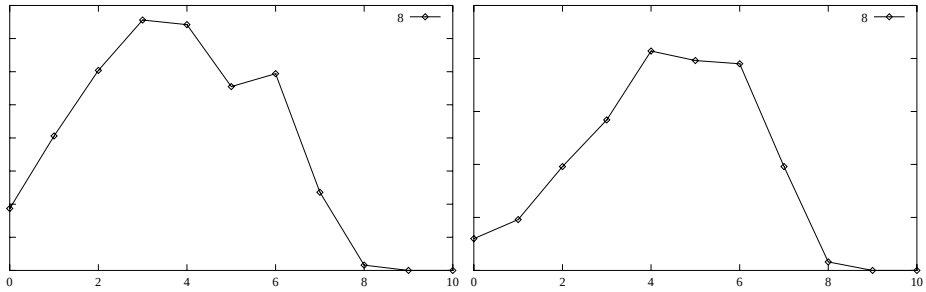


Figure 5.11: *Distributions of differential DC values of* `lena.pgm` *(left) and* `pepper.pgm` *(right) for* Q_1*=8.*

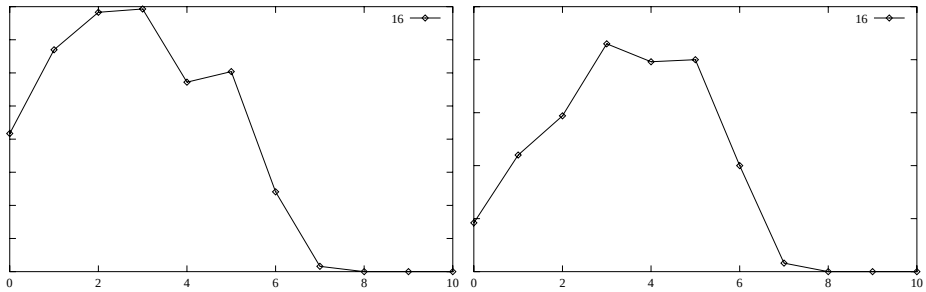


Figure 5.12: *Distributions of differential DC values of* `lena.pgm` *(left) and* `pepper.pgm` *(right) for* Q_1*=16.*

5.8.1 Huffman Table Specifications of Various Images

Huffman Table Specifications and Huffman Code

The following examples show that even if the original image is the same, different quality levels produce different Huffman tables. This shows that finding the Huffman tables of an image is not easy even if the Huffman tables of the same image with different quality levels is known. The examples are the Huffman table specifications of two JPEG images created from `lena.ppm` with quality level 75% and 50 % and the resulting Huffman codewords generated by the algorithm in CCITT Rec.T.81(1992 E) : Annex C, Figure C.1, C.2 and C.3.

Quality level 75 %

DC table 0

- Length = 28
- Table class = 0
- Identifier = 0

The following data is :

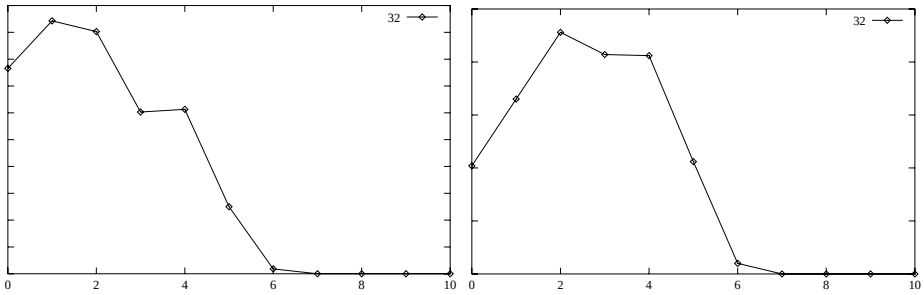


Figure 5.13: *Distributions of differential DC values of* `lena.pgm` *(left) and* `pepper.pgm` *(right) for* Q_1*=32.*

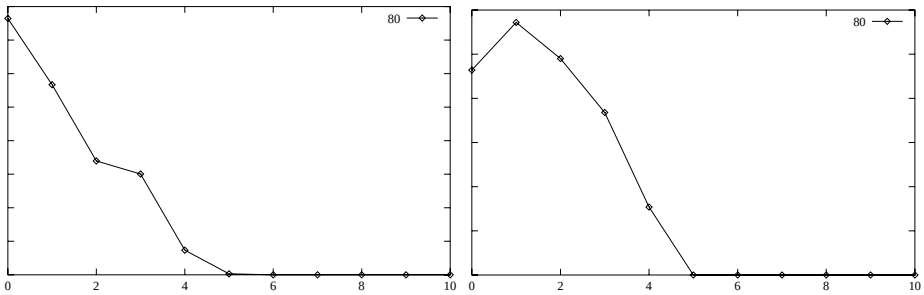


Figure 5.14: *Distributions of differential DC values of* `lena.pgm` *(left) and* `pepper.pgm` *(right) for* Q_1*=80.*

length	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
# of symbols	0	1	5	1	1	1	0	0	0	0	0	0	0	0	0	0
symbols		4	1	7	0	8										
			2													
			3													
			5													
			6													

and the resulting Huffman codewords are :

0:11110 1:010 2:011 3:100 4:00 5:101 6:110 7:1110 8:111110

AC table 0

- Length = 64
- Table class = 1
- Identifier = 0

The following data is :

length	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
# of symbols	0	1	3	3	3	2	4	3	7	3	3	4	3	0	1	5
symbols		1	2	0	5	65	6	50	20	66	7	36	52		114	22
			3	4	18	81	19	113	35	209	21	51	67			38
			17	33	49		34	129	145	225	82	98	130			146
							97		161			241				162
									177							178
									193							
									240							

and the resulting Huffman codewords are :

0:1010 1:00 2:010 3:011 4:1011 5:11010 6:1111000

7:11111111010 17:100 18:11011 19:1111001 20:111110110 21:11111111011

22:1111111111111010 33:1100 34:1111010 35:111110111 36:111111111010

38:1111111111111011 49:11100 50:11111000 51:111111111011 52:1111111111100

65:111010 66:1111111010 67:1111111111101 81:111011 82:11111111100

97:1111011 98:111111111100 113:11111001 114:111111111111100 129:11111010

130:1111111111110 145:111111000 146:1111111111111100 161:111111001

162:1111111111111101 177:111111010 178:1111111111111110 193:111111011

209:1111111011 225:1111111100 240:111111100 241:111111111101

Quality level 50 %

DC table 0

- Length = 27
- Table class = 0
- Identifier = 0

The following data is :

length	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
# of symbols	0	2	3	1	1	1	0	0	0	0	0	0	0	0	0	0
symbols		2	1	0	6	7										
		3	4													
			5													

and the resulting Huffman codewords are :

0:1110 1:100 2:00 3:01 4:101 5:110 6:11110 7:111110

AC table 0

- Length = 60
- Table class = 1
- Identifier = 0

The following data is :

length	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
# of symbols	0	1	4	1	3	3	3	2	4	5	3	3	4	2	3	0
symbols		1	0	33	4	5	19	50	20	6	51	36	21		67	37
			2		18	65	34	113	35	66	193	98	114		83	52
			3		49	81	97		129	82	209	240	225			162
			17						145	161			241			
										177						

and the resulting Huffman codewords are :
0:010 1:00 2:011 3:100 4:11010 5:111010 6:1111111000
17:101 18:11011 19:1111010 20:111111000 21:1111111111010 33:1100
34:1111011 35:111111001 36:111111111010 37:111111111111100 49:11100
50:11111010 51:11111111010 52:111111111111101 65:111011 66:1111111001
67:11111111111100 81:111100 82:1111111010 83:11111111111101 97:1111100
98:111111111011 113:11111011 114:1111111111011 129:111111010 145:111111011
161:1111111011 162:111111111111110 177:1111111100 193:11111111011
209:11111111100 225:1111111111100 240:111111111100 241:1111111111101

Huffman Table Specifications for `lena`

Quality=95%

Quantization table

```
 2   1   1   1   1   1   2   1
 1   1   2   2   2   2   2   4
 3   2   2   2   2   5   4   4
 3   4   6   5   6   6   6   5
 6   6   6   7   9   8   6   7
 9   7   6   6   8  11   8   9
10  10  10  10  10   6   8  11
12  11  10  12   9  10  10  10
```

DC table

length	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
# of symbols	0	1	5	1	1	1	1	1	0	0	0	0	0	0	0	0
symbols		5	3	9	2	1	0	10								
			4													
			6													
			7													
			8													

AC table

length	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
# of symbols	0	1	3	2	4	3	6	4	3	7	3	3	3	2	2	11
symbols		1	2	4	0	7	8	129	20	9	36	22	67	23	162	68
			3	5	6	49	19	145	50	21	82	51	114	52	37	146
			17		18	65	34	161	177	35	241	98	130			10
					33		81	240		66						24
							97			193						39
							113			209						83
										225						99
																115
																131
																178
																194

Quality=75%

Quantization table

```
 8   6   6   7   6   5   8   7
 7   7   9   9   8  10  12  20
13  12  11  11  12  25  18  19
15  20  29  26  31  30  29  26
28  28  32  36  46  39  32  34
44  35  28  28  40  55  41  44
48  49  52  52  52  31  39  57
61  56  50  60  46  51  52  50
```

DC table

length	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
# of symbols	0	1	5	1	1	1	0	0	0	0	0	0	0	0	0	0
symbols		4	1	7	0	8										
			2													
			3													
			5													
			6													

AC table

length	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
# of symbols	0	1	3	3	3	2	4	3	7	2	6	1	5	0	1	5
symbols		1	0	4	5	65	6	50	20	82	7	98	37		130	22
			2	17	18	81	19	113	35	209	21		52			38
			3	33	49		34	129	66		36		67			83
							97		145		51		114			146
									161		225		241			178
									177		240					
									193							

Quality=50%

Quantization table

```
 16  11  12  14  12  10  16  14
 13  14  18  17  16  19  24  40
 26  24  22  22  24  49  35  37
 29  40  58  51  61  60  57  51
 56  55  64  72  92  78  64  68
 87  69  55  56  80 109  81  87
 95  98 103 104 103  62  77 113
121 112 100 120  92 101 103  99
```

DC table

length	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
# of symbols	0	2	3	1	1	1	0	0	0	0	0	0	0	0	0	0
symbols		2	1	0	6	7										
		3	4													
			5													

AC table

length	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
# of symbols	0	1	4	1	3	3	3	2	3	7	3	4	0	7	1	0
symbols		1	0	33	4	5	19	50	35	6	98	36		21	162	
			2		18	65	34	113	129	20	193	114		37		
			3		49	81	97		145	51	209	225		52		
			17							66		240		67		
										82				83		
										161				146		
										177				241		

Quality=25%

Quantization table

```
 32  22  24  28  24  20  32  28
 26  28  36  34  32  38  48  80
 52  48  44  44  48  98  70  74
 58  80 116 102 122 120 114 102
112 110 128 144 184 156 128 136
174 138 110 112 160 218 162 174
190 196 206 208 206 124 154 226
242 224 200 240 184 202 206 198
```

DC table

length	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
# of symbols	0	2	3	1	1	0	0	0	0	0	0	0	0	0	0	0
symbols		1	0	5	6											
		2	3													
			4													

AC table

length	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
# of symbols	0	2	2	1	3	3	3	2	5	3	3	3	5	1	0	0
symbols		0	2	3	4	18	34	19	5	51	82	20	52	67		
		1	17		33	65	50	113	35	161	193	36	114			
					49	81	97		66	177	209	98	225			
									129				240			
									145				241			

Quality=10%

Quantization table

```
 80  55  60  70  60  50  80  70
 65  70  90  85  80  95 120 200
130 120 110 110 120 245 175 185
145 200 255 255 255 255 255 255
255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255
```

DC table

length	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
# of symbols	0	3	1	1	1	0	0	0	0	0	0	0	0	0	0	0
symbols		0	3	4	5											
		1														
		2														

AC table

length	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
# of symbols	1	1	0	2	2	2	1	5	0	1	3	4	3	1	1	0
symbols	0	1		2	33	3	18	34		4	66	19	98	82	193	
				17	49	65		50			129	35	161			
								81			145	51	225			
								97				177				
								113								

Huffman Table Specifications for `pepper`

Quality=95%

Quantization table

2	1	1	1	1	1	2	1
1	1	2	2	2	2	2	4
3	2	2	2	2	5	4	4
3	4	6	5	6	6	6	5
6	6	6	7	9	8	6	7
9	7	6	6	8	11	8	9
10	10	10	10	10	6	8	11
12	11	10	12	9	10	10	10

DC table

length	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
# of symbols	0	2	2	3	1	1	1	1	0	0	0	0	0	0	0	0
symbols		6	5	3	2	1	10	0								
		7	8	4												
				9												

AC table

length	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
# of symbols	0	1	3	2	4	4	4	4	3	6	5	2	5	4	3	0
symbols		1	2	4	6	0	19	9	50	21	22	225	10	24	53	
			3	5	18	7	34	20	145	35	36	241	23	67	83	
			17		33	8	81	113	161	66	82		37	115	210	
					49	65	97	129		177	98		51	130		
										193	209		114			
										240						

Quality=75%

Quantization table

8	6	6	7	6	5	8	7
7	7	9	9	8	10	12	20
13	12	11	11	12	25	18	19

```
15  20  29  26  31  30  29  26
28  28  32  36  46  39  32  34
44  35  28  28  40  55  41  44
48  49  52  52  52  31  39  57
61  56  50  60  46  51  52  50
```

DC table

length	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
# of symbols	0	2	3	1	1	1	1	0	0	0	0	0	0	0	0	0
symbols		4	2	7	1	0	8									
		5	3													
			6													

AC table

length	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
# of symbols	0	1	3	2	4	3	5	4	7	5	5	7	5	0	0	0
symbols		1	2	0	5	6	19	20	7	21	22	37	38			
			3	4	18	65	34	50	35	36	114	51	68			
			17		33	81	97	145	66	82	130	67	115			
					49		113	161	177	98	146	99	131			
							129		193	225	241	116	132			
									209			162				
									240			194				

Quality=50%

Quantization table

```
16  11  12  14  12  10  16  14
13  14  18  17  16  19  24  40
26  24  22  22  24  49  35  37
29  40  58  51  61  60  57  51
56  55  64  72  92  78  64  68
87  69  55  56  80 109  81  87
95  98 103 104 103  62  77 113
121 112 100 120  92 101 103  99
```

DC table

length	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
# of symbols	0	2	3	1	1	1	0	0	0	0	0	0	0	0	0	0
symbols		3	1	6	0	7										
		4	2													
			5													

AC table

length	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
# of symbols	0	1	3	3	3	1	5	5	5	6	2	10	3	1	0	0
symbols		1	0	4	5	65	19	6	20	51	21	36	37	130		
			2	17	18		34	50	35	66	114	52	67			
			3	33	49		81	129	177	82		68	83			
							97	145	193	178		98				
							113	161	209	225		99				
										240		115				
												131				
												146				
												147				
												241				

Quality=25%

Quantization table

```
 32  22  24  28  24  20  32  28
 26  28  36  34  32  38  48  80
 52  48  44  44  48  98  70  74
 58  80 116 102 122 120 114 102
112 110 128 144 184 156 128 136
174 138 110 112 160 218 162 174
190 196 206 208 206 124 154 226
242 224 200 240 184 202 206 198
```

DC table

length	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
# of symbols	0	2	3	1	1	0	0	0	0	0	0	0	0	0	0	0
symbols		2	1	0	6											
		4	3													
			5													

AC table

length	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
# of symbols	0	1	4	1	3	2	4	2	9	3	3	5	1	0	0	0
symbols		1	0	33	4	65	5	50	20	35	36	52	240			
			2		18	81	19	113	66	51	83	67				
			3		49		34		82	193	98	115				
			17				97		114			130				
									129			209				
									145							
									146							
									161							
									177							

Quality=10%

Quantization table

```
 80  55  60  70  60  50  80  70
 65  70  90  85  80  95 120 200
130 120 110 110 120 245 175 185
145 200 255 255 255 255 255 255
255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255
```

DC table

length	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
# of symbols	0	3	1	1	0	0	0	0	0	0	0	0	0	0	0	0
symbols		0	3	4												
		1														
		2														

AC table

length	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
# of symbols	0	2	2	1	3	3	3	4	3	1	1	0	0	0	0	0
symbols		0	2	3	18	65	34	4	35	161	66					
		1	17		33	81	97	19	51							
					49	113	145	50	129							
								82								

Summary of Experiments

From the results of the above experiments, it can be seen that the Huffman tables of the two images which are generated from the same image, will be different if different quality levels are used and the tables of one image cannot be used for the other image. If Huffman code is unknown, it is known that recovering information from the Huffman-coded data is not easy. For methods to recover information from coded data without knowing the Huffman code, further research is required.

5.8.2 Conclusion

From above, the most effective part to be encrypted will be the Huffman table specifications. The specifications vary not only with images but also with quality levels as shown in Section 5.8 and 5.6.2. The failure of the re-construction of the Huffman tables in the decoder is fatal for the correct decoding. The advantages of this method are *i*) the size of data to be encrypted is very small (less than 1 % of the JPEG stream in Table 5.7) and so it is computationally inexpensive, and by using a well-established encryption algorithm, it can provide high level of security, *ii*) the encryption and the decryption of the Huffman table specifications can be applied directly to the JPEG stream without decoding it and so already existing JPEG files can be easily encrypted, and *iii*) the structure of the stream conforms to the JPEG standard and the stream is still recognized as a JPEG stream.

An image viewer (`xv`) used for the experiments in Section 5.7.6, recognizes the encrypted file as a JPEG stream and produces the error message for the Huffman specifications. Since the size of data to be encrypted is very small and the Huffman table varies with images, this scheme provides the computationally inexpensive secure encryption of JPEG streams.

Chapter 6

Wavelet Compression and Encryption

6.1 Introduction

Wavelet compression is a recently developed compression method for digital images. It achieves very high compression with a reasonably high image quality. Various wavelet image compression systems have been proposed in the last decade [4, 90, 91, 7]. Wavelet compression is also used as the basis of a new international standard for image compression, JPEG2000, proposed by ISO/IEC [43].

To prevent unauthorized access to image data, the data needs to be encrypted. Although images are compressed for efficient storage and transmission, generally the compressed data is still large and so applying conventional encryption to the compressed data is computationally expensive. To reduce the computational cost, there are two approaches : *i*) *selective encryption* and *ii*) *elementary cryptographic operations*. In this chapter, we propose encryption schemes for the Discrete Wavelet Transform (DWT) [64] that fall into the latter approach and use random permutation lists.

Firstly, we examine a method using random permutation lists for a DWT-based compression system and then present a scheme for JPEG2000. Finally, we conclude our results.

6.2 Encryption with Discrete Wavelet Transform

We use a simple key dependent transformation: a family of permutations indexed by the key, on wavelet coefficients. The decoder applies the inverse permutation before processing the coefficients. An unauthorized user who does not know the key cannot recover the image because the correct order of coefficients is not known. The amount of masking will depend on the number of subbands to which the permutation is applied.

In this section we investigate the security and efficiency of the above system using

a specific implementation [22]. We will show that this basic system can provide various degrees of masking of information and the transformation does not have a drastic effect on the compression ratio. To assess the security of the system we will examine possible attacks. In particular, we demonstrate a plaintext attack that uses a number of well chosen transform coefficients to derive the secret permutation. We argue that if higher security is required, then a block cipher algorithm can be used to mask the coefficients of the lowest subband while permutations are used for the other subbands.

This section is organized as follows. Firstly, we briefly review the wavelet compression system used in our experiments and then in Section 6.2.2 present the encryption scheme use a random permutation. In Section 6.2.3 and 6.2.4 we present our attack and using block encryption to enhance security. In Section 6.2.5 and 6.2.6 we show the results of the experiments and in Section 6.2.7 we conclude.

6.2.1 Wavelet Image Compression

In a wavelet compression system an image is decomposed into *subbands* which are represented by real-valued wavelet coefficients. The transform stage is followed by a *quantization* stage which converts the real-valued coefficients to whole numbers. Finally an *entropy coder* is used to compress the output of the quantizer. The transform stage is invertible and the compression is the result of quantization and entropy coding stages. There are numerous approaches to quantization with varying levels of performance.

Transform

The Discrete Wavelet Transform (DWT) consists of a sequence of wavelet filter banks [64]. In the 2-dimensional transform, firstly each pixel row in the image is decomposed into coarse and detailed parts and is down-sampled, and then each column of the coarse and detailed parts is decomposed into coarse and detailed parts and is down-sampled again. This results in four parts: coarse-coarse, coarse-detail (both from the row coarse part), detail-coarse and detail-detail parts (both from row detailed part). The last three parts compose the output subbands while the coarse-coarse part is the input of the next filter bank. Hence each filter bank produces three subbands except for the last filter bank which has four subbands.

Quantization and Entropy Coding

The implementation uses a quantization algorithm [110] which quantizes the wavelet coefficients of each subband independently. The entropy coder is an adaptive arithmetic coder [9].

6.2.2 Encryption Using Random Permutation

Encrypting MPEG coded data using key dependent permutations to permute transform coefficients of the Discrete Cosine Transform (DCT) has been proposed in [107, 95]. In using a permutation of the wavelet transform coefficients, the following points must be taken into account. Firstly, the DCT is used on fixed size ($n \times n$) blocks of pixels in the image and produces the same number of coefficients for each block, while wavelet coefficients are computed for the whole image and so the number of coefficients depends on the image size. Secondly, the quantization precision will be subband dependent and so the number of bits allocated to each coefficient will vary for different subbands. This means that the permutation must be applied to each subband separately, otherwise a large compression rate drop or distortion can be expected.

We use a subband-based permutation system with a different permutation for each subband. Let $V^{(b)} = (v_0^{(b)}, v_1^{(b)}, v_2^{(b)}, \ldots, v_i^{(b)}, \ldots)$ be the wavelet coefficient block in subband b, and $v_i^{(b)}$ denote the ith coefficient in the subband and $|V^{(b)}|$ be the number of coefficients. The permutation can permute all $|V^{(b)}|$ coefficients, or we may break the block into sub-blocks and permute the coefficients in each sub-block. For simplicity we assume the former.

There are $|V^{(b)}|!$ permutations for subband b and so the number depends on the image size and b, i.e. the subband which is permuted. Let $K^{(b)}$ be the key used for generation of the permutation for subband b. Keys can be chosen independently for each subband, or by using a master key $\hat{K}$ and a key generation algorithm that has $\hat{K}$ as input and produces keys for all subbands.

Permutation of coefficients may be implemented in the following two ways.

Case 1 Wavelet coefficients in each subband are permuted after transformation and before quantization.

Case 2 Quantized coefficients in each subband are permuted after quantization and before entropy coding.

The implementation uses an arithmetic coder to encode the quantized coefficients. The coder encodes the i th bits of all coefficients in a subband, starting from the most

significant bits and moving to the least significant bits and produces the entropy coded stream in an embedded manner, that is, the stream consists of number of segments, from the most significant segment to the least significant segment, each of which contains the encoded data.

The two methods could result in different compression rates if the quantization method depends on the order of coefficients (context). For example, with vector quantization, the compression rate in Case 1 will be affected. In both cases, if the entropy coding is sensitive to the order of quantized coefficients then a drop in compression rate will be expected.

6.2.3 Chosen Plaintext Attack

To evaluate the security of the system, firstly we need to find the number of keys. In the above the key determines the permutation and there are $|V^{(b)}|!$ possible permutations for subband b consisting of $|V^{(b)}|$ coefficients and so $\sum_{i=1} n|V^{(i)}|!$ possible permutation for the image which consists of n subbands. For example, a 512×512 gray scale image with four subbands, $|V^{(1)}|$, $|V^{(2)}|$, $|V^{(3)}|$, and $|V^{(4)}|$ are $256 \times 256 = 65536$ and so $\sum_{i=1} 4|V^{(i)}|! = 4 \times 256!$. As can be seen from above, the value can be very large. This ensures that an *exhaustive search attack* will be infeasible. However this would not guarantee security of the system as more efficient attacks could be possible. In the following we describe a chosen plaintext attack that recovers the secret permutation by examining the system output on a number of well chosen attack images.

In a *chosen plaintext attack*, the attacker has access to an encoder with a secret key. He can choose a plaintext, i.e. an image, obtain the corresponding ciphertext and analyze the relationship between the plaintext and ciphertext to gain information about the key. He can repeat this analysis on a number of images. The attack uses the fact that key dependent permutations only change the order of coefficients but do not change the values of coefficients. Hence if the attacker can create an image which produces distinct values for coefficients in a subband, he can find the permutation applied to that subband by capturing the coefficients in his decoder and observing their order.

The assumptions are:

i) The algorithms of transform, quantization, entropy coding and the permutation are known but the key is secret. *ii*) The attacker can obtain ciphertext (compressed images) corresponding to any chosen plaintext (original images). *iii*) The attacker can create an image with coefficients of his choice.

The attack steps are as follows.

1. The attacker generates an image which has distinct coefficients in one or more subbands.
2. He/she gives the image to the target encoder.
3. The encoder transforms, permutes, quantizes and encodes the image.
4. The attacker obtains the encoded image and decodes the compressed image using his/her decoder.
5. The attacker captures the decoded values between the de-quantization and the inverse-transform and compares the values with his/her chosen values and finds the permutation.

A single attack image in general may recover only part of the permutation.

For the attack to be successful care must be taken against the following types of errors.

i) The pixel values that are obtained from the coefficients must be in the valid range (typically 0 to 255). *ii*) The error in converting real-valued coefficients to integer-valued pixels should not destroy the distinctness of coefficients calculated in the decoder. *iii*) The quantization process in the target encoder should not destroy the distinctness of the coefficients.

In [22], calculation of the transform and inverse-transform has considerable precision and so many distinct coefficients can be chosen. The main restriction is due to the quantization precision. If each of the coefficients in subband b is represented by $q^{(b)}$ bits then there can be at most $2^{q^{(b)}}$ distinct coefficients and $2^{q^{(b)}}$ distinct values can reveal the permutation of $2^{q^{(b)}}$ coefficients in a single attack attempt.

This means that coarser subbands are more vulnerable to the attack because, *i*) the quantization precision for the coarser subbands is higher than the detailed ones, and so the number of distinct coefficients that can be used in a single attack will be larger and ii) less number of attack attempts are required for the coarser subbands since coarser subbands have smaller number of coefficients compared with the higher subbands.

6.2.4 Enhancing Security

The chosen plaintext attack is mainly effective for the lowest subband which largely contributes to PSNR in general. To improve the security of the scheme we can use a

traditional block cipher algorithm such as AES [73] to encrypt coefficients in the lowest subband similar to encrypting DC coefficients of a DCT transformed image in [107]. Using AES ensures that the lowest subband is highly secure, and using permutations for the higher subbands ensures that the detail information are hidden too.

It is interesting to note that the cost of block encryption for wavelet based systems is lower than DCT based systems. The following example clarifies this point.

We compare the wavelet transform with the DCT applied to MPEG-1. In MPEG-1, a color image of $m \times n$ pixels is broken into $\frac{m \times n}{16 \times 16}$ macro-blocks, where each of macro-blocks is decomposed into 4 luminance and 2 chrominance 8×8 blocks. This means that the image is represented by $\frac{6mn}{256}$ 8×8 blocks. Each of the 8×8 blocks is DCT-transformed into one 8 bit DC coefficient and 63 AC coefficients. To encrypt the DC coefficients, $\frac{48mn}{256} = 0.1875mn$ bits must be encrypted.

In a wavelet transform with five 2-D filter banks, we assume the 2 chrominance components have half the size of the luminance component (as in the DCT case). In each filter bank, the size of each component is reduced by half in width and height, i.e. 1/4 in total, and $1/4^5$ for 5 filter banks. In this case each of the color components will be independently transformed which results in $\frac{mn}{4^5}$ and $\frac{mn}{4 \times 4^5}$ coefficients in the coarsest luminance and chrominance subbands, respectively. So overall, the transform of the 3 color components produces $\frac{5mn}{4^6}$ coefficients. If the quantization precision for the coefficients is 8 bits, this results in $\frac{40mn}{4^6} \approx 0.0098mn$ bits which is approximately 1/20 of the DCT case.

It is worth noting that the above comparison assumes that the information in DC coefficients of 8×8 DCT is almost the same as that of the coarsest subband of the above wavelet transform. To find the exact amount of information in the two cases (DC coefficients of DCT and the lowest subband of wavelet), a more detailed analysis is required.

6.2.5 Experiments

We used an implementation [22] that uses the Antonini wavelet [4], employs a 2-dimensional transform, has five filter banks and decomposes the image into $3 \times 4 + 4 = 16$ subbands. The program takes the compression rate as an input parameter and adjusts the quantization precisions of subbands to achieve the compression rate. The exact compression rate may not be achievable simply because a one bit change in the precision of subband b results in $|V^{(b)}|$ bits change in the quantizer output size and the output size will be decreased by $|V^{(b)}|$ bits. If the permutation results in a considerable drop in

Table 6.1: Compression rate and PSNR with permuted subbands when the target compression rate is specified to 8:1.

Permuted subband # in encoding	I-permuted subband # in decoding	Comp. rate (bpp)	PSNR
None	None	0.9975	39.448
0	None	0.9975	13.051
1	None	0.9975	20.947
2	None	0.9975	28.620
3	None	0.9975	26.837
4	None	0.9975	24.494
5	None	0.9975	29.814
6	None	0.9975	29.900
7	None	0.9983	26.995
8	None	0.9978	31.359
9	None	0.9978	31.722
10	None	1.0014	29.946
11	None	0.9993	33.460
12	None	0.9995	34.907
13	None	1.0040	34.003
14	None	1.0005	36.746
15	None	0.9982	38.602
0 to 7	None	0.9983	11.634
8 to 15	None	1.0162	24.979
0 to 15	None	1.0168	11.444
0 to 15	0 to 15	1.0168	39.448

the compression then the precisions in some of the subbands will be reduced to achieve the given compression rate and will result in lower quality image and drop in PSNR (Peak Signal to Noise Ratio).

The test image is `lena.pgm`, which is a 512×512 image with 256 level gray scale (8 bits/pixel). The compression rate was set to 8:1, i.e. 1 bit/pixel (bpp).

The results of the experiment are shown in Table 6.1 and Figure 6.1 - 6.3. Firstly the image is encoded and decoded without permutations and PSNR is calculated. Then coefficients in various subbands are permuted and PSNR is calculated. It can be seen that the permutation of coarser subbands results in a larger PSNR drop compared to the detailed part. In all the above cases, the drop in compression rate is less than 2%.

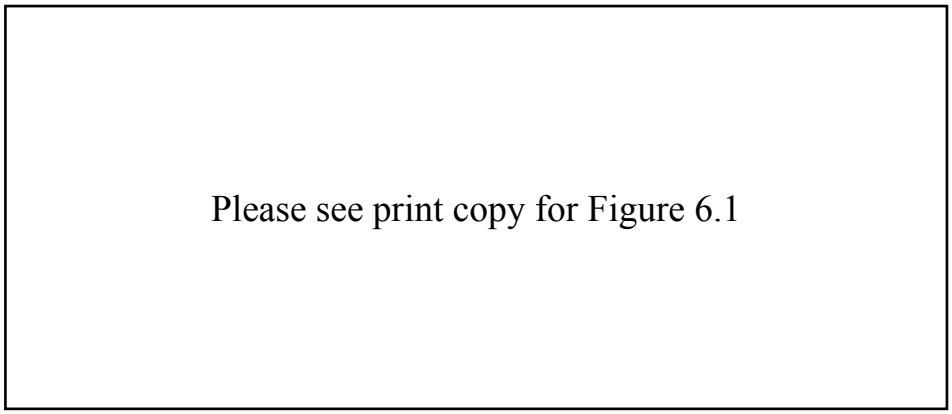


Figure 6.1: The original image (left) and the recovered image without inverse-permutations when the image is encoded with subband 0 permuted (right).

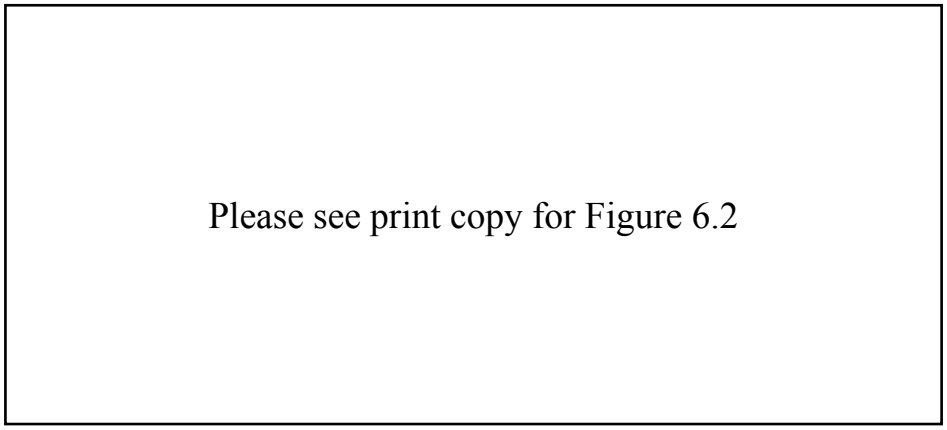


Figure 6.2: The recovered image without inverse-permutations when the image is encoded with subband 15 permuted (left) and the recovered image without inverse-permutations when the image is encoded with subbands 0 to 15 permuted (right).

6.2.6 Compression Rate

It can be seen from the experiment results the drop in compression rate is small. This is mainly due to the type of the quantization and entropy coding algorithms: the quantization algorithm processes one coefficient at a time and does not depend on other coefficients in the subband, and the entropy coder uses context information which is orthogonal to the direction of the permutation. The number of permuted subbands does not change the precision of the quantization process because the drop in the compression rate is not large enough to necessitate a change in the precision. In the experiment the permutations were applied between the transform and the quantization but similar results can be expected if the permutations are used after the quantization

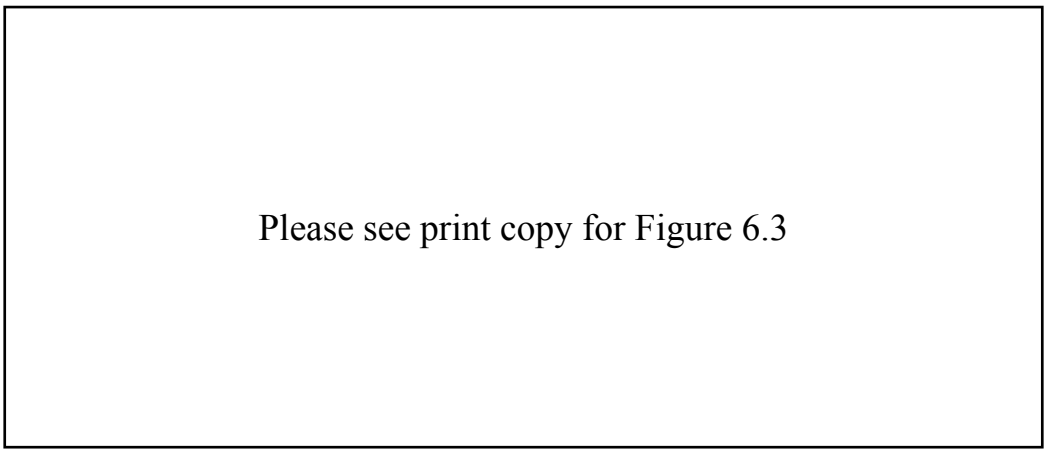


Figure 6.3: The recovered image without inverse-permutations when the image is encoded with subbands 0 to 7 permuted (left) and the recovered image without inverse-permutations when the image is encoded with subbands 8 to 15 permuted (right).

because of the above reasons. The permutation has a small impact on the compression rate in the implementation.

6.2.7 Concluding Remarks

We have shown that permuting one or a small number of subbands in a wavelet based compression system can add security without having a large effect on the compression rate. By increasing the number of subbands with permuted coefficients the security can be increased and so the system provides variable levels of security. We have shown that despite reasonable perceptual masking of the information and the very large size of the key space, the system is vulnerable to a chosen plaintext attack in which a specially constructed image is encoded and the output of the decoder is analyzed. The attack is particularly effective against the lowest subband. We proposed an extension of the system that provides protection against this attack.

6.3 A JPEG2000 Encryption System

JPEG2000 is a new image compression international standard proposed by ISO/IEC [43]. It uses the Discrete Wavelet Transform (DWT) for its transform and has a number of advantages over JPEG [45]. In particular it provides *i*) better rate-distortion performance at low bit-rates, *ii*) better lossless and lossy compression in a single codestream, *iii*) support for images with size larger than $2^{16} \times 2^{16}$ pixels (this is the maximum size for JPEG), *iv*) simple single decompression architecture, compared to 44 modes

in JPEG decoders, *v*) error resiliency, *vi*) better image quality for computer generated images compared to JPEG, and *vii*) better image quality for bi-level images [44].

In Chapter 5, we considered possible ways of incorporating encryption into the JPEG compression system. Not all proposed methods can be extended to JPEG2000. In particular it is not possible to use the selective encryption of JPEG Huffman table specifications described in Chapter 5. This is because in JPEG, the Huffman table specifications are critical for correct decoding and must be known before decoding starts. JPEG2000 uses an adaptive arithmetic coder. The coder starts from a fixed known model and updates the model adaptively according to the input symbols. Since the initial model is public, the entropy coder cannot be used for encryption. Instead, we apply elementary cryptographic operations on transform coefficients to encrypt JPEG2000 data. We propose an encryption scheme using random permutation lists for JPEG2000. The objective of the scheme is to encrypt image data without significant drop in compression performance and conform to the JPEG2000 image compression standard. Using computationally expensive encryption algorithms on image data with large size will reduce the coding efficiency. Our proposed scheme avoids the drop in compression performance by using a simple cryptographic operation. The scheme provides various degrees of image masking by allowing the choice of subbands and bit-planes that are encrypted.

We first review the JPEG2000 compression system and then describe the proposed encryption scheme and show our experiment results. We analyze security of the scheme against chosen-coefficient attack described in Section 6.3.3 and finally give concluding remarks.

6.3.1 JPEG2000 Compression System

The JPEG 2000 encoding procedure is decomposed into three stages:

1. Transformation
2. Quantization
3. Embedded Block Coding with Optimized Truncation (*EBCOT*) [108, 109]

Transformation

In the transformation stage, pixels are transformed into wavelet coefficients in subbands using the Discrete Wavelet Transform (DWT). JPEG 2000 provides two types

of transforms: *i*) an integer wavelet transform that is invertible, and *ii*) a real number wavelet transform that is not invertible. Their transforms produce integer and real number coefficients respectively. For lossless compression, the invertible integer wavelet transform must be used.

Quantization

After the transformation stage, each of the subbands is divided into rectangular blocks called *code-blocks*.

In the quantization stage, the wavelet coefficients in a code-block is divided by a quantization step size and the decimal part is truncated. The quantization step size for each code-block is determined by the required image quality and bit-rate.

A quantized coefficient is represented by its sign and magnitude, that is, a sign bit and the absolute value of the coefficient.

Embedded Block Coding with Optimized Truncation (EBCOT)

In the EBCOT stage, each code-block is independently encoded into the bit-stream in such a way that the more important information always precedes less important information. This is the heart of the embedded bit-stream organization. The bit-stream is organized using a *Coefficient Bit Modeler (CBM)* . Using the CBM, the coefficients in a code-block are encoded using an adaptive binary arithmetic coder.

Code-block

A code-block is a rectangular block of sign-magnitude pairs representing quantized coefficients. A pair consists of a sign bit and an absolute value of a coefficient. Each bit layer of the binary representation of the absolute values forms a *bit-plane*. For example, the rectangular block of the most significant bits of the absolute values forms the most significant bit-plane.

Significance state

Each coefficient in the code-block has an associated binary state variable called *significance state*. The significance state of a coefficient is initialized to 0 and becomes 1 when the bit in a bit-plane is `1` during the encoding of bit-planes from the *Most Significant Bit* (MSB) to the *Least Significant Bit* (LSB).

In the following, we describe the procedure of coefficient bit modeling.

- The bit-planes are encoded from the MSB to the LSB.

- Starting from the most significant bit-plane, the number of consecutive bit-planes consisting of all zeros is recorded in the header of the compressed data. No coding is made for these bit-planes.
- Once a non-zero value in a bit-plane is detected, all the subsequent bit-planes are coded. The encoding of a bit-plane consists of three coding passes.
  1. *Significance propagation*
  2. *Magnitude refinement*
  3. *Cleanup*

To encode a bit-plane, it is divided into groups of 1×4 bits and the groups are scanned from left to right and from top to bottom.

Let B be a binary $m \times n$ matrix representing an $m \times n$ bit-plane.

$$B = \begin{pmatrix} b_{1,1} & b_{2,1} & \dots & b_{m,1} \\ b_{1,2} & b_{2,2} & \dots & b_{m,2} \\ \vdots & & \dots & \vdots \\ b_{1,n} & b_{2,n} & \dots & b_{m,n} \end{pmatrix} . \tag{6.1}$$

Then group g_k of four bits is given by $g_k = (b_{i,j}, b_{i,j+1}, b_{i,j+2}, b_{i,j+3})$ where $1 \leq i \leq m$, $j \in \{1, 5, 9, 13, ..., n-3\}$ and $k = i + \frac{j-1}{4}m$. In the three coding passes, all g_k in the bit-plane are scanned in the order of $k = 1, 2, 3, ..., \frac{mn}{4}$.

In the following section, the details of the three coding passes are described.

Coding Passes

The three coding passes take place in the following manners.

1. For the first bit-plane that includes a non-zero bit, *cleanup pass* is executed.
2. Once the first bit-plane including non-zero bit is encoded using the *cleanup pass* (the above case), the three passes take place in the following order.
   (a) Significance propagation
   (b) Magnitude refinement
   (c) Cleanup

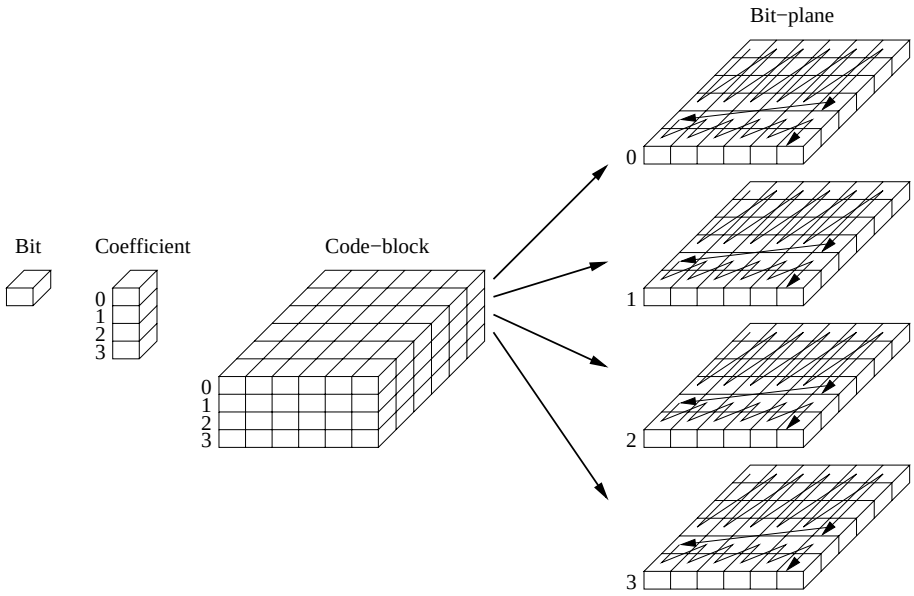


Figure 6.4: Code-block and bit-planes : A quantized coefficient consists of bits and A code-block consists of $m \times n$ quantized coefficients. The ith bit-plane is the collection of ith significant bits of the $m \times n$ quantized coefficients. The bits in a bit-plane are scanned as shown by the arrows.

These passes produce a sequence of pairs of *decision* and *context* that are passed to the arithmetic coder. There are 10 *contexts*.

Significance propagation The significance propagation includes only the coefficients of significance state = 0 and non-zero *context*. In this pass, the *context* is obtained from the neighbors of the coefficient. The *decision* is the bit of each coefficient.

Magnitude refinement The magnitude refinement pass includes the coefficients of significance state = 1 except the coefficients whose significance state has changed from 0 to 1 in the immediately proceeding significance propagation pass. In this pass, the *context* is obtained from the neighbors of the coefficient. The *decision* is the bit of each coefficient.

Cleanup The cleanup includes all the coefficients with the significance state = 0 and *Context 0*. In this pass, the groups of four bits are run-length coded. The *decisions* are the result of the run-length coding of the four bits and the *UNIFORM context* is used to encode the *decisions*.

Adaptive Binary Arithmetic Coder

The entropy-coder used in JPEG 2000 is an adaptive binary arithmetic coder. The input of the arithmetic coder is a sequence of pairs of *decision* and *context* produced by the CBM. The arithmetic coder consists of *i*) 10 models, which represent the probability distribution of binary source symbols corresponding to 10 *contexts* and a context used in the run-length coding, and *ii*) a coder, which encodes source symbols based on the source symbol probability distribution given by the models corresponding to the *contexts*.

The arithmetic coder encodes the *decision* based on the model of the corresponding *context*. The encoding is done by dividing the current interval into two sub-intervals according to the probabilities of the binary symbols, and choosing one of the sub-intervals as a new interval. The choice of the intervals is determined by the binary symbol to be encoded. There are 9 flags corresponding to the 9 *contexts* and each of them indicates which binary symbol is the *Less Probable Symbol* (LPS), i.e. the binary symbol which has smaller probability.

A model is assigned to each of the 9 *contexts*. A model holds the probability of the LPS. The probability is approximated by one of the forty six values given in [44] and the index to one of the forty six values is kept in the model. This means that the adaptive model is order-0, that is, the probabilities of symbols are determined by how frequently each symbol appears in the input data without taking into account of its preceding symbols. After encoding a *decision*, the corresponding model is updated.

6.3.2 Encryption Using Random Permutation Lists

For an image to be correctly decoded, the encoding order of the coefficients must be known to the decoder. If the encoding order of coefficients is secret then the decoder cannot correctly decode the image. The scheme described herein realizes this by using random permutation lists. The coding order is determined by random permutation lists that are generated using a *secret key*. The method such as [71, 72] can be used although this thesis does not cover the method of generating random permutation lists in details.

The JPEG2000 encryption takes place in the EBCOT stage. The random permutation list encryption mechanism is inserted into the coefficient bit modeler subsystem so as to minimize the impact of the encryption on the compression rate.

In the JPEG 2000 standard specification, in the three coding passes the groups of

4 coefficients g_k in a bit-plane are scanned in the order of $k = 1, 2, 3, ..., \frac{mn}{4}$, i.e. from left to right from top to bottom. To add encryption to this stage, groups are scanned based on random permutation lists generated from the secret key.

Let Π be a permutation of integers $1, 2, 3, ..., \frac{mn}{4}$. Then the scanning order of groups specified by JPEG2000 can be represented by $(1, 2, 3, ..., \frac{mn}{4})$, and after using permutation will become $\Pi_1, \Pi_2, \Pi_3, ..., \Pi_{\frac{mn}{4}}$. That is, the order of scanning the groups will be $(g_{\Pi_1}, g_{\Pi_2}, g_{\Pi_3}, ..., g_{\Pi_{\frac{mn}{4}}})$. For example, $\Pi = (14, 3, 7, ..., 5)$ shows that the groups are scanned as $g_{14}, g_3, g_7, ..., g_5$. The encoder and the decoder scan the groups of the coefficients in the order specified by S_{rand}.

A different scanning order is used for each bit-plane of each code-block. That is, every bit-plane has its unique scanning order. The same secret key K must produce the same set of Π for all bit-planes in encoding and decoding.

6.3.3 Security of JPEG2000 Encryption

In this section we examine security of the proposed JPEG2000 encryption system. Firstly, we briefly review existing encryption systems for MPEG using random permutation lists and the attacks against these systems, and then investigate whether or not these attacks are effective on the JPEG 2000 encryption system. Finally we examine the effectiveness of the chosen-coefficient attack in Section 6.2.

For MPEG, the systems in [107] and [95] use random permutation lists to permute DCT coefficients in blocks. These systems are known to be insecure [2, 114] because of the following reasons.

1. It is known that lower frequency coefficients carry larger energy and so coefficient values of the lower frequencies are larger than those of higher frequencies. Because of this energy distribution, the original coefficient order in a block can be roughly recovered by sorting the coefficients.

2. If known images are encrypted, it is possible to find the permutation by comparing DCT coefficients of the known images and those of the permuted ones.

In the wavelet case, the reason 1 above is not applicable because coefficients in a subband, having the same frequency component, are permuted. Permutation of the coefficients will hide the local energy distribution over the entire image. Since the energy distribution over regions of images varies by image, sorting coefficients will not be an effective attack against the permutation of wavelet coefficients. To address reason 2 above, we examine the effectiveness of the chosen-coefficient attack.

Chosen Coefficient Attack

The assumptions are as follows.

- The secret is the random permutation lists that determine the scanning order of bit-planes.
- The attacker has access to the decoder with the key loaded and can conduct repeated experiments on the decoder.

The chosen coefficient attack is defined as follows. Let v be a w dimensional input vector of coefficients, F_k be a key dependent permutation and F_k^{-1} be the inverse of F_k. The ciphertext $u = F_k(v)$ is also a w dimensional vector.

The attacker chooses C which consists of distinct values, $c_1, c_2, ..., c_w$ where $c_a \neq c_b$, $\forall a, b \in \{1, 2, ..., w\}$, $a \neq b$. Then he calculates $P_k^{-1}(C)$, that is, decodes C using the decoder with the key k loaded and obtains $V' = P_k^{-1}(C)$. By analyzing C and V', the attacker can find the permutation P_k^{-1} and its inverse P_k.

If the number of the possible values of c_a is w' and $w' < w$, then the attacker chooses w' distinct values and constructs $C = (c_1, c_2, ..., c_{w'-1}, c_{w'}, c_{w'}, ..., c_{w'})$ for one experiment. This will reveal the permutation of $w' - 1$ coefficients, and $\frac{w}{w'-1}$ times repetition of this experiment will reveal the whole permutation.

In JPEG2000 encryption system, groups of 4 bits are randomly scanned. Let D be a set of distinct binary vectors of length 4. Then a group of 4 bits can be represented by a vector in D. D is given as follows.

$$D = \{(0,0,0,0), (0,0,0,1), (0,0,1,0), (0,0,1,1), \ldots, (1,1,1,1)\} \tag{6.2}$$

and the number of vectors w in D is $w = 2^4 = 16$.

Let d be a subset of D and w' as the number of vectors in d. The encryption algorithm permutes groups of 4 bits, i.e. a set of binary vectors of length 4. If the groups are a subset of D, i.e. the groups are d, then the permutation of w' vectors can be traced. From above, if the attacker constructs a compressed image so that it consists of a set of distinct vectors, then the permutation of the vectors can be traced from the decoded image.

For the attack, first the attacker constructs a JPEG 2000 compressed attack image consisting of bit-planes, each of which includes the vectors in D. Let x be the number of groups in a bit-plane. Then the method to construct an attack image is as follows.

	Algorithm 1 : Construction of a compressed attack image.
1:	For each code-block
2:	For each bit-plane
3:	If $x \leq w$
4:	Construct the bit-plane using x vectors in D.
5:	Else
6:	Construct the bit-plane so that
7:	one vector in D appears $x - w + 1$ times
8:	and the other $w - 1$ vectors in D appear once.

Next, the attacker decodes the attack image using the decoder with the secret key and obtains the decoded image.

Then he transforms the decoded image and obtains the coefficients. The coefficients are quantized to obtain the same representation as the bit-planes. The vectors in bit-planes are inverse-permuted from the attack image and then the permutation of distinct vectors is found.

We note that if $x \leq w$, the permutation of all groups can be found. However, if $x > w$, the permutation of $w-1$ groups can be found. In practice, it will be reasonable to assume that $x > w$. For example, the default value of x in the implementation [36] is 1024.

Assuming $x > w$, if the same permutation lists are used for all the bit-planes in a code-block, the attacker can choose vectors such that the $w-1$ distinct vectors in each bit-plane covers different regions so that each bit-plane can reveal $w-1$ permutations. This single experiment can reveal $l(w-1)$ permutations where l is the number of bit-planes in a $m \times n$ code-block. The cost of the attack for a code-block is as follows.

$$\epsilon_{same} = \frac{mn}{4l(w-1)} . \tag{6.3}$$

The attacker needs to repeat the experiment ϵ_{same} times to find the permutation for a $m \times n$ code-block.

If different permutation lists are used for the bit-planes, the attacker can find the permutation of $w-1$ vectors for each bit-plane in one attempt and the number of attempts required to reveal the permutation lists used for all bit-planes is as follows.

$$\epsilon_{diff} = \frac{mn}{4(w-1)} . \tag{6.4}$$

The total cost depends on the image size and the number of filter banks. Let Λ be the number of $m \times n$ blocks in the image and Θ denote the number of filter banks. Then

the number of blocks in a subband of the output of the jth filter bank is approximately $\frac{\Lambda}{(2^j)^2}$ where $1 \leq j \leq \Theta$. The total number of blocks Ξ is as follows.

$$\Xi \approx \sum_{j=1}^{\Theta} 3\frac{\Lambda}{(2^j)^2} + \frac{\Lambda}{(2^\Theta)^2} = \Lambda \ . \tag{6.5}$$

For each attempt, the attacker obtains same amount of information about the permutation lists.

For example, when $m = 64$ and $n = 64$, and the number of bit-planes l in a code-block is $l = 10$, $\epsilon_{same} = \frac{1024}{10(16-1)} = 6.83 \approx 2^{2.8}$ and we have $\epsilon_{diff} = \frac{1024}{16-1} = 68.27 \approx 2^6$.

Impact of the Adaptive Arithmetic Coder on the Attack

The chosen-coefficient attack works only if the correctly inverse-permuted coefficients are obtained. To obtain the correctly inverse-permuted coefficients, the compressed attack image must be correctly decoded by the decoder to be attacked. However, as described in Section 6.3.1 the adaptive arithmetic coder is unable to correctly decode the compressed attack image without knowing the random permutation lists.

The arithmetic coder uses more than one adaptive model and the models are chosen by *contexts* and so if the order of *contexts* in the encoder and the decoder does not match, the encoder and the decoder will lose synchronization. The order of *contexts* is determined by the scanning order of 4 bit groups in bit-planes and the scanning order is chosen by the random permutation lists. Hence, the order of *contexts* cannot be found without knowing the random permutation lists.

To make the decoder successfully recover the coefficients, the attacker needs to exhaustively experiment the permutation of pairs of *decision* and *context* when he constructs the compressed attack image.

Let n_i be the number of occurrences of *context* i in encoding of the whole image. Then $\sum_{i=1}^{10} n_i$ *decision* and *context* pairs are encoded by the arithmetic coder. If the attacker tries all possible order of the pairs, the number of trials N will be

$$N = \frac{(\sum_{i=1}^{10} n_i)!}{n_1!n_2!...n_{10}!} \tag{6.6}$$

which is very large and so the attack is impractical.

6.3.4 Experiments

In this section we show the results of our experiments on the JPEG2000 encryption system. The implementation of the system is based on the JPEG2000 codec, JasPer-0.072 [36]. For each color component, subband, coefficient block and bit-plane, a different permutation was used. The images used in the experiments were `lena.ppm`, `mandril.ppm`, `peppers.ppm`. All image sizes were 512×512. The compression rate was set to 32:1. 50 trials of encryption were done for each of the images using different keys to observe the dependency of the compression rate and the quality of encrypted image on a secret key. For the encryption and the decryption, different secret key was used. Firstly, the encryption was applied to different subbands. All the bit-planes in the chosen subbands were encrypted. The sets were $\{0\}$, $\{7\}$, $\{13\}$, $\{1,2,3\}$, $\{7,8,9\}$, $\{13,14,15\}$. Then one of bit-plane 0 (the most significant bit-plane), 1 and 2 was chosen for encryption of the subband sets $\{1,2,3\}$, $\{7,8,9\}$, $\{13,14,15\}$.

Figure 6.5, 6.6, 6.7, 6.8, 6.9, 6.10, and 6.11 show the encrypted images of `lena.ppm`, `mandoril.ppm`, and `peppers.ppm` encrypting subbands $\{0\}$, $\{7\}$, $\{13\}$, $\{1,2,3\}$, $\{7,8,9\}$, $\{13,14,15\}$, and all subbands (0 to 15), respectively.

If the lower subbands were encrypted, the image showed large color spots. A spot corresponds to a coefficient in the encrypted subband. Encrypting the middle subbands showed noise pattern similar to moire and encrypting the highest subband showed very small impact on the quality of the decompressed image. Encrypting the lower subbands resulted in the lower PSNR but if higher subbands were intact, the edges can be clearly seen.

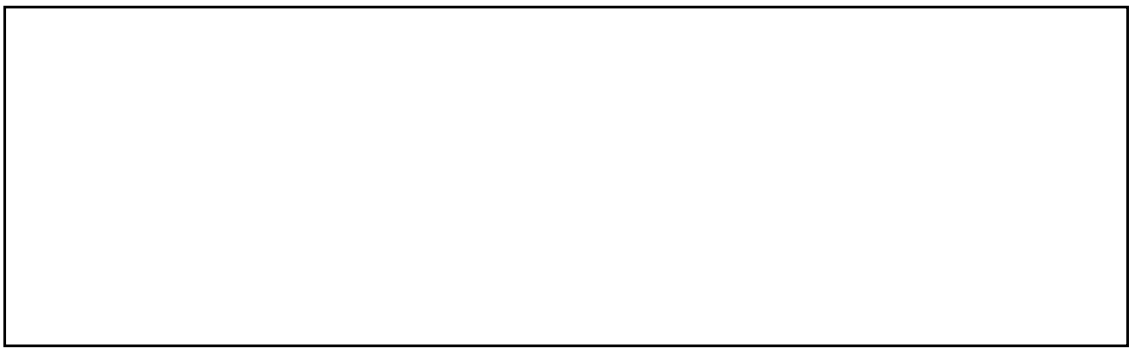


Figure 6.5: Encrypting subband 0 : `lena.ppm` (left), `mandoril.ppm` (middle) and `peppers.ppm` (right). The color spots correspond to low subband coefficients. The encryption decreased the image quality but the details (i.e. edges) are visible.

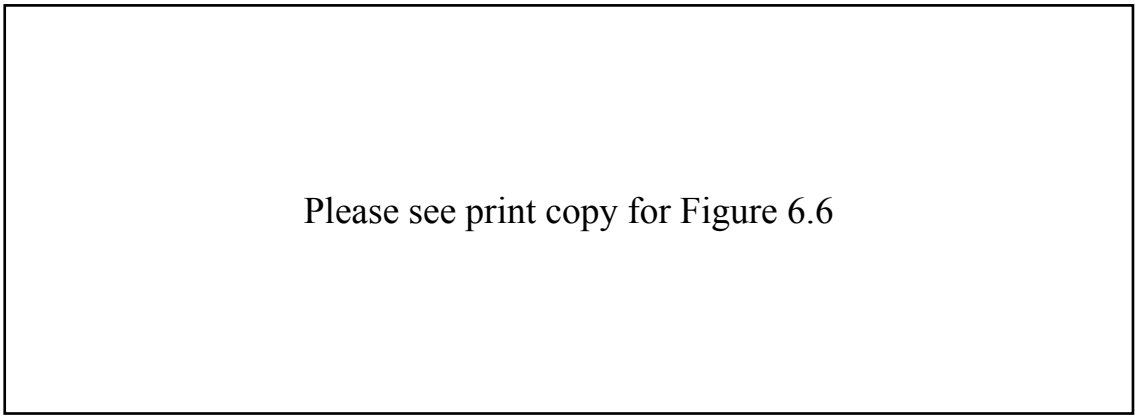


Figure 6.6: Encrypting subband 7 : `lena.ppm` (left), `mandoril.ppm` (middle) and `peppers.ppm` (right). The encryption decreased the quality less compared to encrypting low subbands. The images are recognizable.

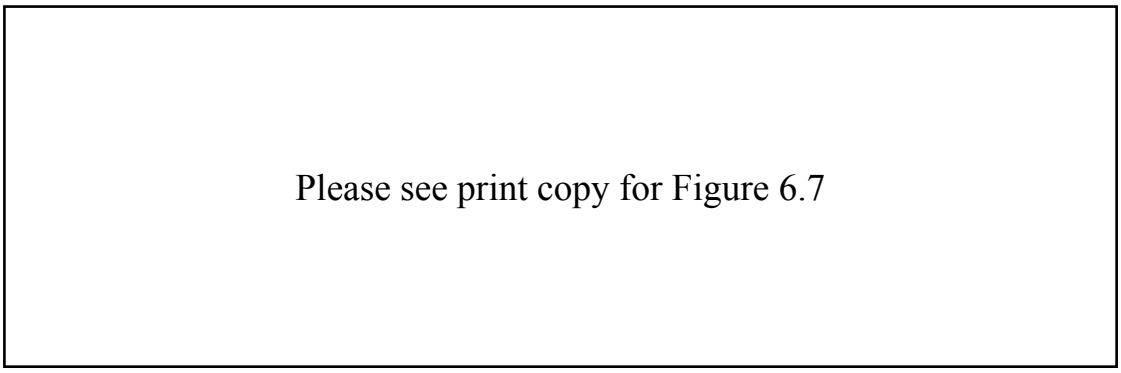


Figure 6.7: Encrypting subband 13 : `lena.ppm` (left), `mandoril.ppm` (middle) and `peppers.ppm` (right). Some noise can be found in the active regions but the encryption did not decrease the quality very much. The images are similar to the original ones.

Figure 6.12, 6.13, 6.14, 6.15, 6.16, 6.17, 6.18, 6.19, and 6.20 shows the encrypted images of `lena.ppm`, `mandoril.ppm`, and `peppers.ppm` encrypting bit-plane 0 of subbands $\{1, 2, 3\}$, bit-plane 1 of subbands $\{1, 2, 3\}$, bit-plane 2 of subbands $\{1, 2, 3\}$, bit-plane 0 of subbands $\{7, 8, 9\}$, bit-plane 1 of subbands $\{7, 8, 9\}$, bit-plane 2 of subbands $\{7, 8, 9\}$, bit-plane 0 of subbands $\{13, 14, 15\}$, bit-plane 1 of subbands $\{13, 14, 15\}$, and bit-plane 2 of subbands $\{13, 14, 15\}$, respectively.

From the images, it can be seen that the more significant bit-plane of the lower subband has larger impact on the image quality than the less significant bit-plane of the higher subband.

Table 6.2 shows the ratio of the encrypted file sizes to the compressed file size without encryption. The columns labeled "Average", "Minimum", "Maximum" and "Std. dev." show the average, minimum, maximum and standard deviation of the ratios, respectively, over 50 trials.

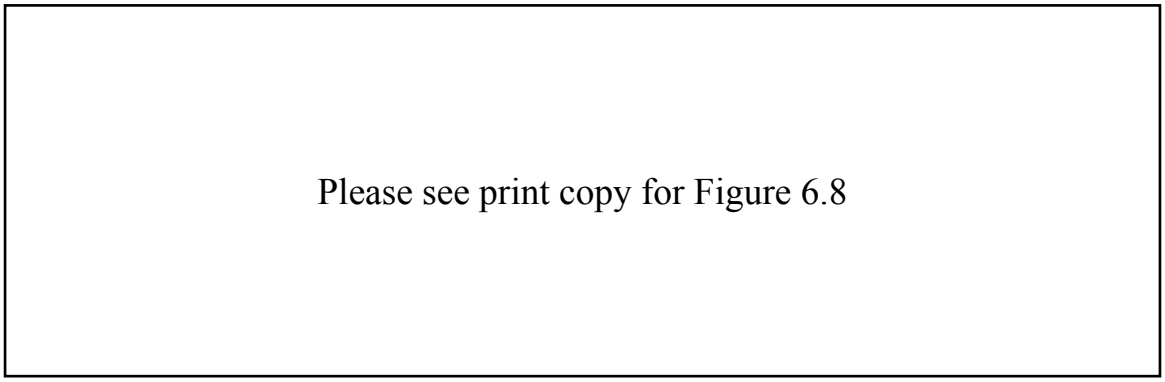


Figure 6.8: Encrypting subband 1, 2, and 3 : `lena.ppm` (left), `mandoril.ppm` (middle) and `peppers.ppm` (right). The quality drop due to the encryption is large but the edges are visible.

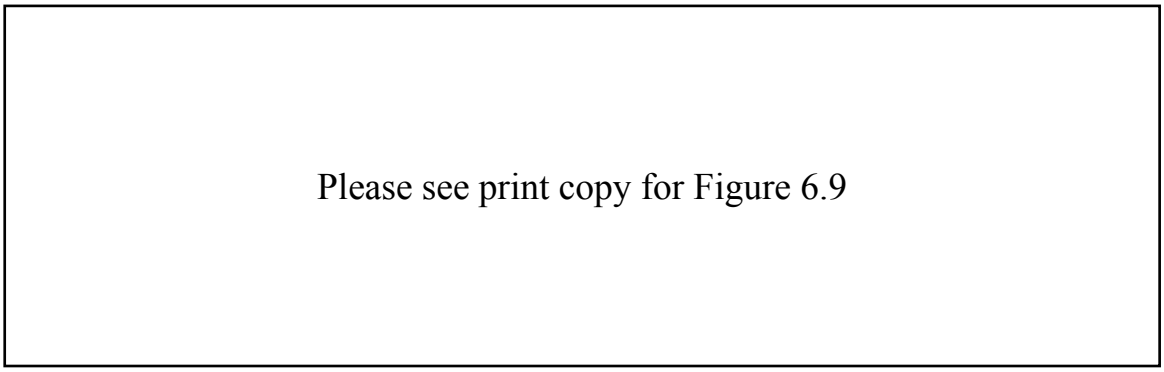


Figure 6.9: Encrypting subband 7, 8, and 9 : `lena.ppm` (left), `mandoril.ppm` (middle) and `peppers.ppm` (right). The encryption has a similar effect to "oil painting". It may be visually disturbing but the images remain recognizable.

Table 6.3 shows the PSNRs of the decrypted images using wrong keys. The columns labeled "Average", "Minimum", "Maximum" and "Std. dev." show the average, minimum, maximum and standard deviation of PSNRs, respectively, over 50 trials.

The graphs in Figure 6.21 (page 149) show the frequencies of each *context* when images were compressed with and without encryption. The graphs in Figure 6.22 (page 150) show frequencies of pairs of *context* and *decision* when images were compressed with and without encryption. Only 5 contexts appeared in the encoding.

6.3.5 Compression Rate

To provide the various degrees of masking, subbands and bit-planes can be selectively encrypted. Selective encryption of image regions can be achieved by selecting specific code-blocks for encryption. Hence the scheme provides flexibility in both in terms of

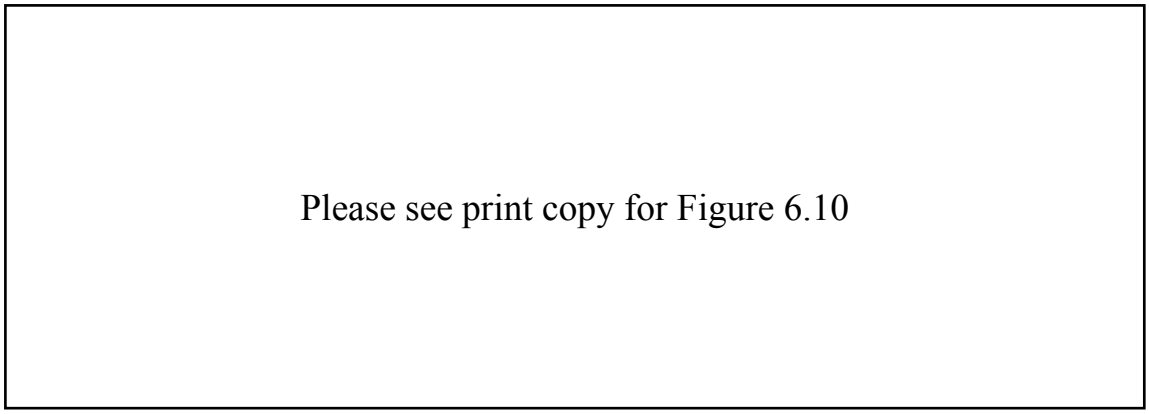


Figure 6.10: Encrypting subband 13, 14, and 15 : `lena.ppm` (left), `mandoril.ppm` (middle) and `peppers.ppm` (right). Some noise can be found in the active regions but the quality drop is small.

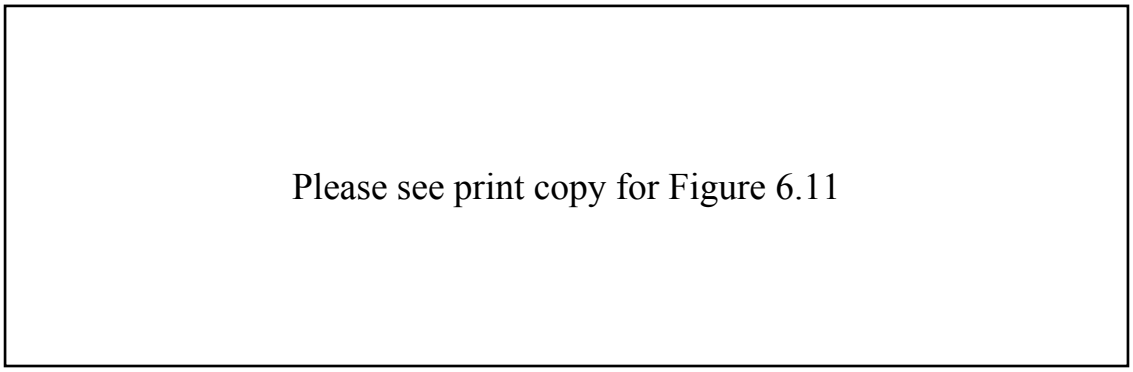


Figure 6.11: Encrypting all subbands (0 to 15) : `lena.ppm` (left), `mandoril.ppm` (middle) and `peppers.ppm` (right). The images are not comprehensible.

the level of masking and region of protection.

In the following, we analyze the impact of permuting the scanning order on the compression rate. JPEG2000 system uses an adaptive binary arithmetic coder of order-0. For such arithmetic coders, the order of input symbols has little impact on the compression rate as long as the size of the input data is large and the probabilities of symbols do not largely change through the encoding. By changing the scanning order, the correlation between an input symbol and its preceding symbols can be destroyed but it is not taken into account in order-0 arithmetic coders. Regardless of the order of the symbols, the model converges into the probability distribution of the input source during encoding.

From the results of our experiments in Section 6.3.4, it can be seen that replacing the original JPEG2000 scan by the random scan does not change the frequencies of *context-decision* pairs very much. This means that the random scan will change the order of

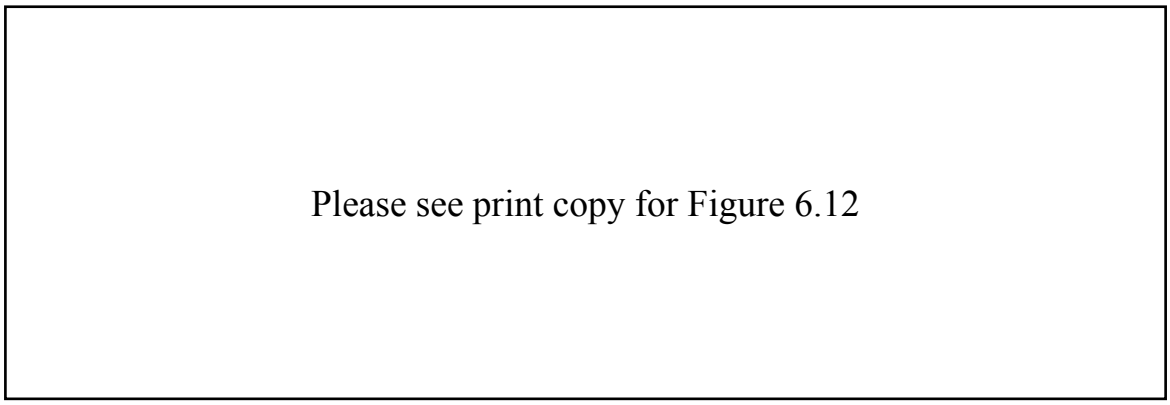


Figure 6.12: Encrypting bit-plane 0 of subbands 1, 2 and 3 : `lena.ppm` (left), `mandoril.ppm` (middle) and `peppers.ppm` (right).

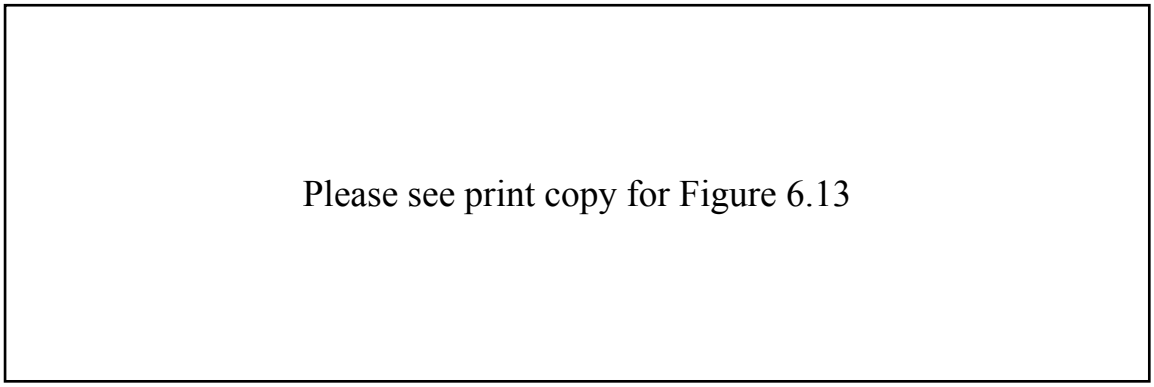


Figure 6.13: Encrypting bit-plane 1 of subbands 1, 2 and 3 : `lena.ppm` (left), `mandoril.ppm` (middle) and `peppers.ppm` (right).

context-decision pairs but does not change their frequencies much. Since changing order has little impact on order-0 adaptive arithmetic coders, the compression rate drop is very small. The random scan does not change the scanning order of 4 bits in each group and so this will also contribute in minimizing the compression rate drop.

The compression rate of the arithmetic coder is approximately 10 % and so its contribution to the entire compression is small. Even if the compression rate drops in the arithmetic coder part, its impact on the overall compression rate will be small.

6.3.6 Concluding Remarks

We proposed a JPEG 2000 compression and encryption scheme using random permutation lists. The tests and simulation results indicate that it provides a simple mechanism of adding encryption to JPEG2000 without significantly degrading the compression performance. The cost of a chosen-coefficient attack against the system is large (equation (6.6)) and so the system is resistant against the attack although the resistance to

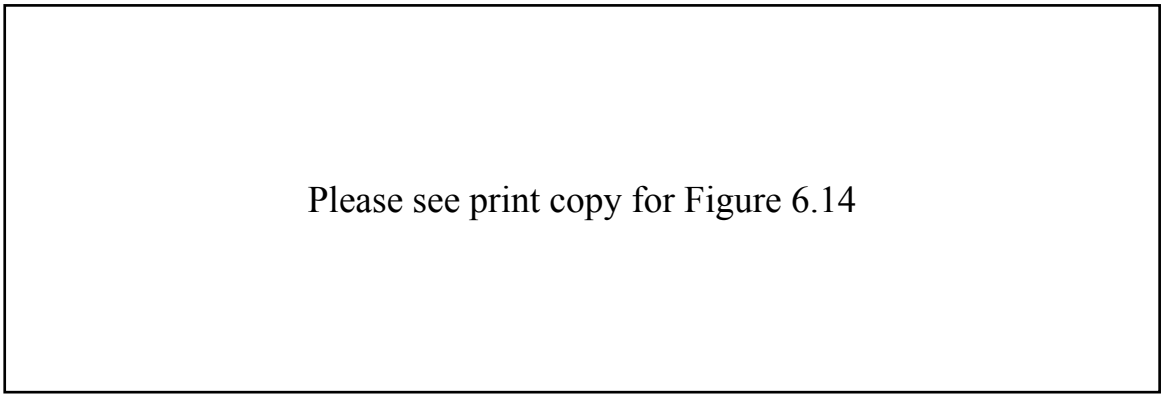


Figure 6.14: Encrypting bit-plane 2 of subbands 1, 2 and 3 : `lena.ppm` (left), `mandoril.ppm` (middle) and `peppers.ppm` (right).

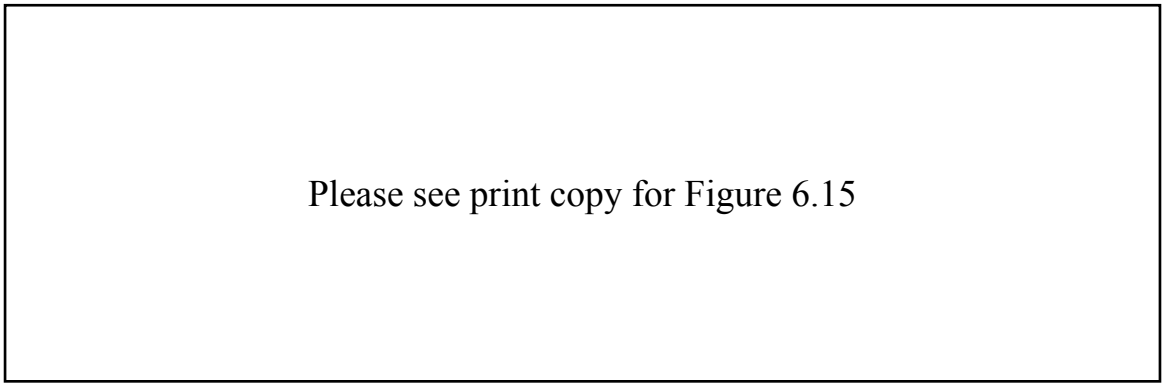


Figure 6.15: Encrypting bit-plane 0 of subbands 7, 8 and 9 : `lena.ppm` (left), `mandoril.ppm` (middle) and `peppers.ppm` (right).

specific attacks is not a guarantee of security.

6.4 Conclusion

We presented two schemes for a specific implementation by Geoff Davis [22] and JPEG2000 image compression system using elementary cryptographic operations, that is, random permutation lists. We showed that if designed it carefully, then permutation can have little influence on the compression rate. Compression systems exploit the correlation among data. Permuting data can destroy this correlation and results in compression rate drops. To avoid this, the permutation was applied before the entropy coding which does not depend on the order of data.

We examined the chosen coefficient attack and showed that in JPEG2000 encryption the attack is ineffective. We note that for the sorting coefficient attack, there is a fundamental difference between the random scan of DWT coefficients and that of DCT

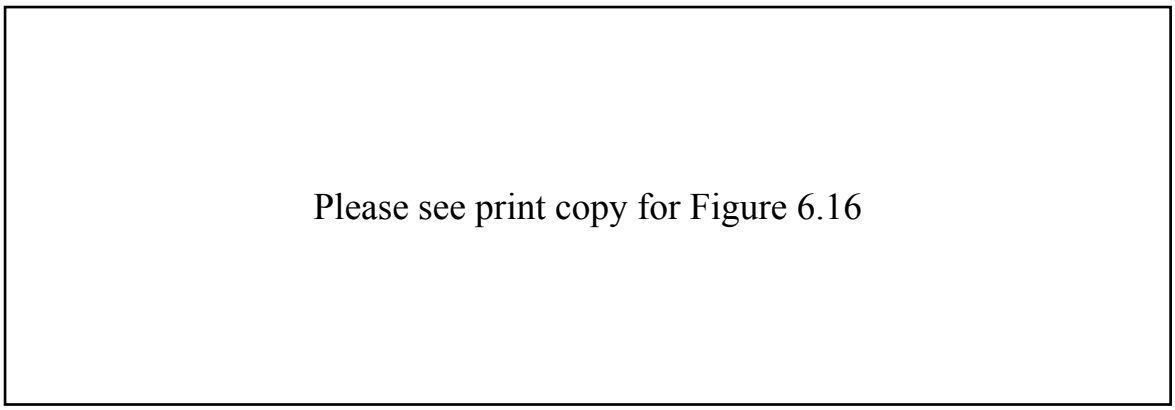


Figure 6.16: Encrypting bit-plane 1 of subbands 7, 8 and 9 : `lena.ppm` (left), `mandoril.ppm` (middle) and `peppers.ppm` (right).

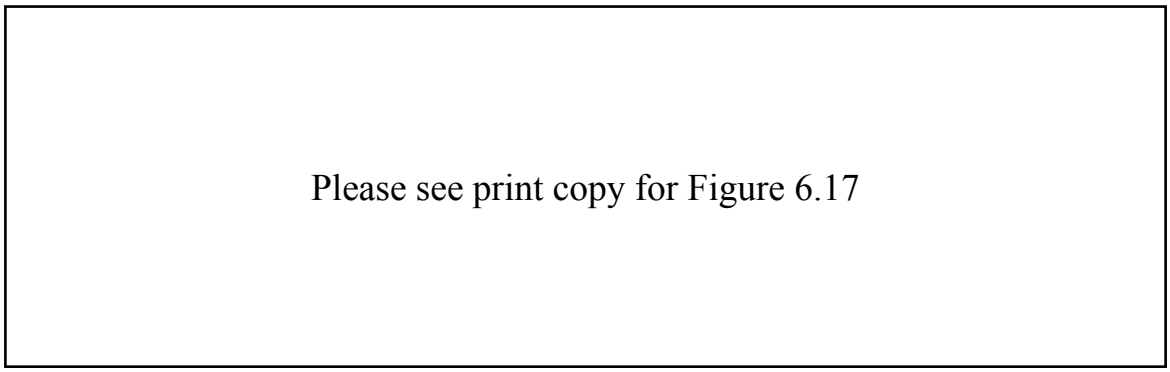


Figure 6.17: Encrypting bit-plane 2 of subbands 7, 8 and 9 : `lena.ppm` (left), `mandoril.ppm` (middle) and `peppers.ppm` (right).

coefficients as described in [107] and [95]. In the DCT case, the permutation of 64 DCT coefficients in a block can be easily reconstructed by sorting the coefficients. This is because the value of a lower frequency coefficient is larger than that of a higher frequency. In the wavelet case, coefficients in a subband of the same frequency are permuted. The permutation of the coefficients will hide the local energy distribution of the image and since the distribution depends on the image, image recovery is more difficult than the DCT case. Hence the sorting coefficient attack is ineffective against wavelet coefficient permutation. For other types of attacks further research is required.

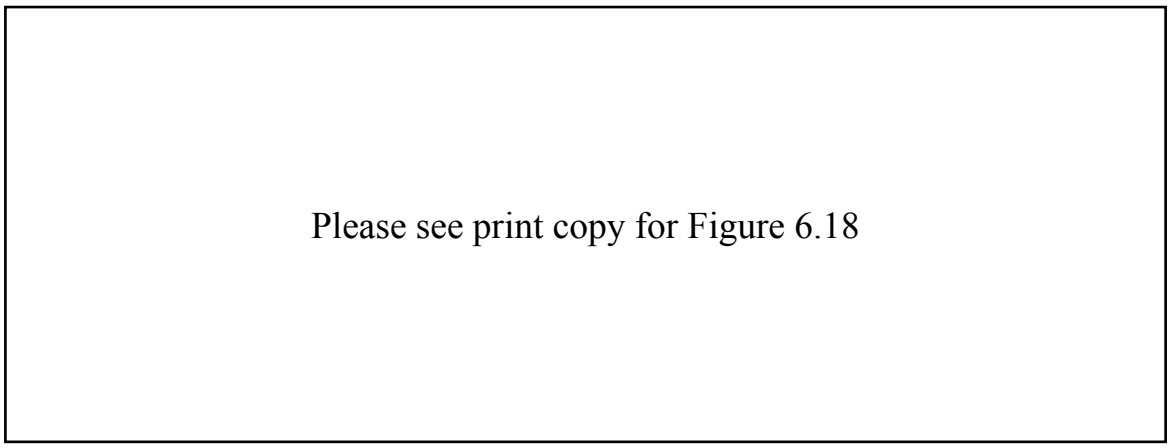


Figure 6.18: Encrypting bit-plane 0 of subbands 13, 14 and 15 : `lena.ppm` (left), `mandoril.ppm` (middle) and `peppers.ppm` (right).

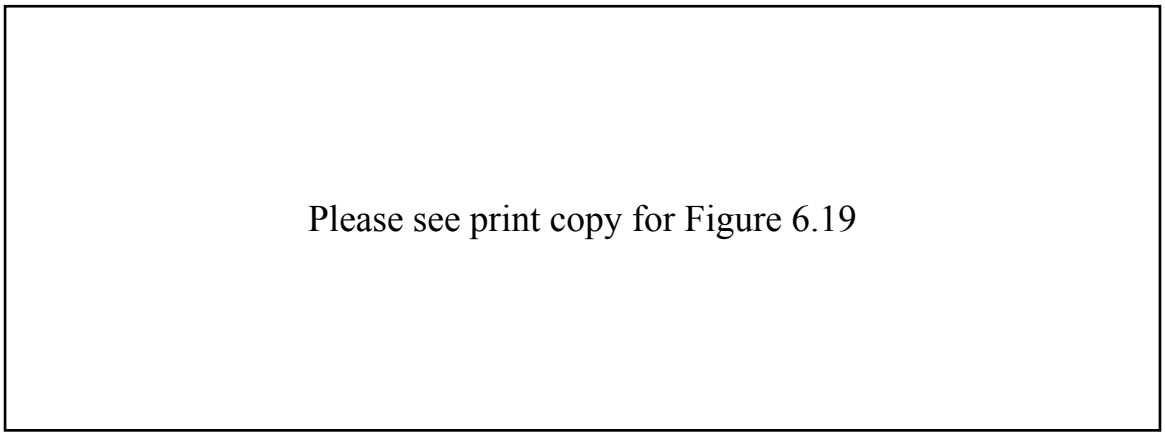


Figure 6.19: Encrypting bit-plane 1 of subbands 13, 14 and 15 : `lena.ppm` (left), `mandoril.ppm` (middle) and `peppers.ppm` (right).

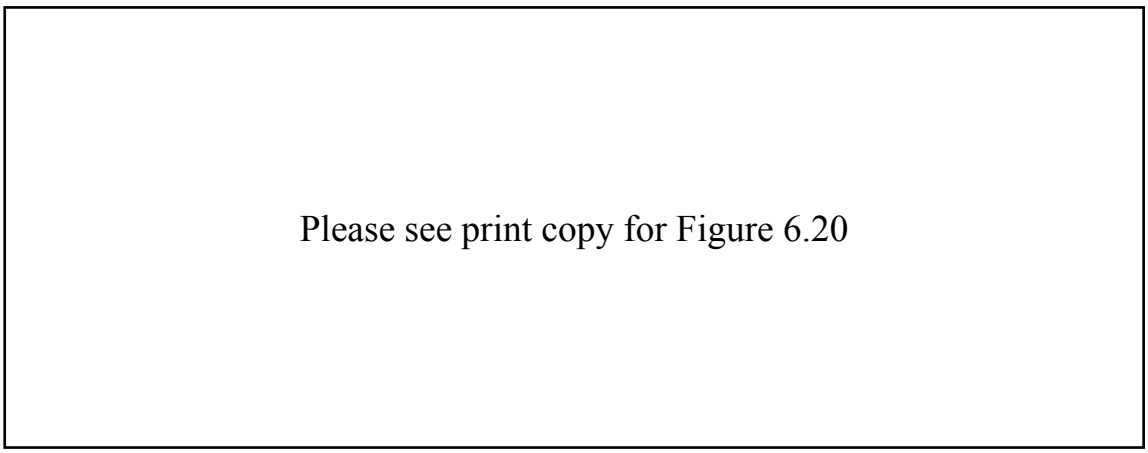


Figure 6.20: Encrypting bit-plane 2 of subbands 13, 14 and 15 : `lena.ppm` (left), `mandoril.ppm` (middle) and `peppers.ppm` (right).

Table 6.2: Compressed file sizes of the random permutation list encryption.

Bit-plane	Subband	Image	Average	Minimum	Maximum	Std. dev.
All	0	`lena`	1.0005	0.9996	1.0007	0.0002
		`mandril`	1.0004	1.0001	1.0006	0.0001
		`peppers`	1.0003	1.0001	1.0007	0.0001
	7	`lena`	0.9998	0.9960	1.0007	0.0010
		`mandril`	1.0003	0.9994	1.0059	0.0015
		`peppers`	1.0000	0.9974	1.0010	0.0011
	13	`lena`	0.9996	0.9980	1.0007	0.0008
		`mandril`	1.0028	1.0011	1.0043	0.0010
		`peppers`	0.9995	0.9975	1.0010	0.0010
	1,2,3	`lena`	1.0000	0.9994	1.0003	0.0002
		`mandril`	1.0018	0.9994	1.0059	0.0029
		`peppers`	0.9999	0.9995	1.0004	0.0002
	7,8,9	`lena`	0.9999	0.9954	1.0007	0.0007
		`mandril`	1.0007	0.9996	1.0056	0.0016
		`peppers`	0.9985	0.9974	1.0009	0.0012
	13,14,15	`lena`	0.9998	0.9986	1.0007	0.0006
		`mandril`	1.0033	0.9994	1.0059	0.0022
		`peppers`	0.9979	0.9896	1.0010	0.0026
Bit 0	1,2,3	`lena`	0.9999	0.9995	1.0002	0.0002
		`mandril`	1.0001	0.9994	1.0059	0.0012
		`peppers`	0.9999	0.9996	1.0003	0.0002
	7,8,9	`lena`	1.0000	0.9996	1.0005	0.0002
		`mandril`	1.0000	0.9996	1.0003	0.0002
		`peppers`	1.0005	0.9975	1.0009	0.0005
	13,14,15	`lena`	0.9995	0.9965	1.0007	0.0011
		`mandril`	1.0025	0.9979	1.0058	0.0018
		`peppers`	0.9996	0.9956	1.0009	0.0011
Bit 1	1,2,3	`lena`	1.0000	0.9995	1.0003	0.0002
		`mandril`	1.0001	0.9995	1.0003	0.0002
		`peppers`	0.9998	0.9995	1.0003	0.0002
	7,8,9	`lena`	1.0002	0.9995	1.0007	0.0005
		`mandril`	1.0006	1.0001	1.0058	0.0008
		`peppers`	0.9982	0.9974	1.0010	0.0011
	13,14,15	`lena`	0.9987	0.9954	1.0007	0.0014
		`mandril`	1.0040	0.9995	1.0059	0.0020
		`peppers`	0.9997	0.9967	1.0010	0.0010
Bit 2	1,2,3	`lena`	1.0000	0.9995	1.0003	0.0002
		`mandril`	1.0003	0.9994	1.0059	0.0019
		`peppers`	0.9998	0.9994	1.0002	0.0002
	7,8,9	`lena`	1.0003	0.9995	1.0007	0.0003
		`mandril`	1.0001	0.9997	1.0006	0.0002
		`peppers`	0.9990	0.9974	1.0010	0.0016
	13,14,15	`lena`	0.9997	0.9969	1.0007	0.0008
		`mandril`	1.0023	0.9988	1.0059	0.0021
		`peppers`	0.9992	0.9974	1.0009	0.0010

Table 6.3: PSNRs of decrypted images using wrong secret keys.

Bit-plane	Subband	Image	Average	Minimum	Maximum	Std. dev.
All	0	`lena`	10.5	8.6	11.9	0.66
		`mandril`	9.5	8.5	11.1	0.63
		`peppers`	9.3	8.5	10.4	0.47
	7	`lena`	20.1	17.2	25.4	2.30
		`mandril`	16.9	15.4	18.8	0.90
		`peppers`	20.9	16.6	27.5	2.72
	13	`lena`	25.8	24.6	27.4	0.60
		`mandril`	18.3	17.6	18.7	0.26
		`peppers`	20.3	17.9	23.6	1.23
	1,2,3	`lena`	17.0	14.7	19.5	1.00
		`mandril`	14.8	13.5	15.9	0.56
		`peppers`	14.3	11.5	16.5	1.14
	7,8,9	`lena`	17.8	15.8	23.3	1.45
		`mandril`	14.4	12.9	16.1	0.73
		`peppers`	17.5	14.9	20.7	1.45
	13,14,15	`lena`	24.8	23.0	25.8	0.52
		`mandril`	14.6	13.8	15.8	0.40
		`peppers`	16.8	15.5	18.5	0.59
Bit 0	1,2,3	`lena`	16.9	15.1	19.5	1.02
		`mandril`	14.4	12.7	15.5	0.64
		`peppers`	14.3	12.0	16.3	0.96
	7,8,9	`lena`	17.6	15.4	20.6	1.18
		`mandril`	14.1	12.7	16.2	0.65
		`peppers`	16.5	13.8	19.3	0.96
	13,14,15	`lena`	24.2	23.1	25.5	0.50
		`mandril`	14.2	13.5	14.8	0.28
		`peppers`	16.4	15.1	17.7	0.58
Bit 1	1,2,3	`lena`	20.6	19.1	22.5	0.67
		`mandril`	16.9	16.2	17.5	0.31
		`peppers`	17.6	15.6	19.3	0.83
	7,8,9	`lena`	23.0	20.0	24.7	0.99
		`mandril`	17.4	16.6	18.0	0.30
		`peppers`	21.8	20.1	24.1	0.81
	13,14,15	`lena`	28.9	28.3	29.6	0.26
		`mandril`	17.8	17.5	18.1	0.15
		`peppers`	24.3	22.2	26.2	1.03
Bit 2	1,2,3	`lena`	24.5	23.6	25.0	0.33
		`mandril`	18.3	18.1	18.5	0.09
		`peppers`	21.5	20.0	22.7	0.55
	7,8,9	`lena`	25.3	24.3	26.0	0.35
		`mandril`	18.2	18.0	18.4	0.09
		`peppers`	24.5	23.0	25.3	0.45
	13,14,15	`lena`	29.9	29.7	30.2	0.11
		`mandril`	18.9	18.8	18.9	0.03
		`peppers`	29.0	27.6	29.8	0.52

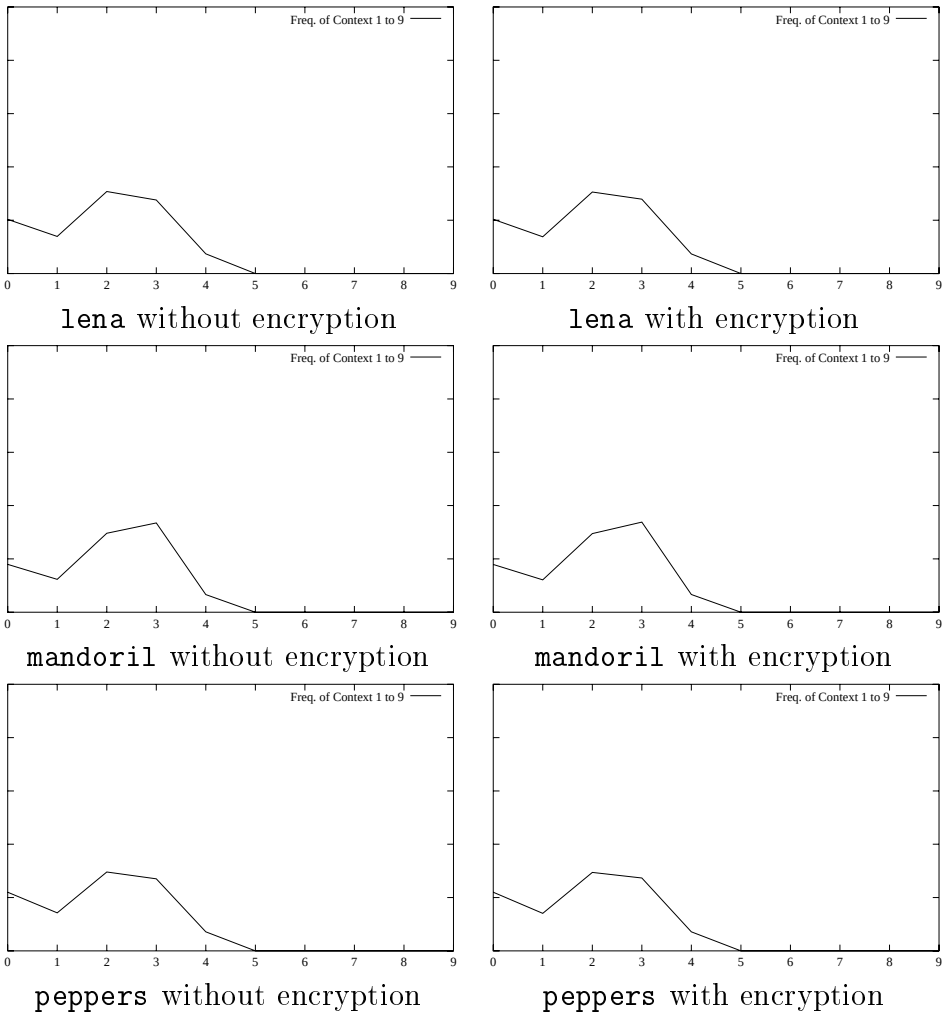


Figure 6.21: Frequencies of 10 *contexts* in the encoding of `lena.ppm`, `mandoril.ppm` and `peppers.ppm` without encryption (left column) and with encryption (right column).

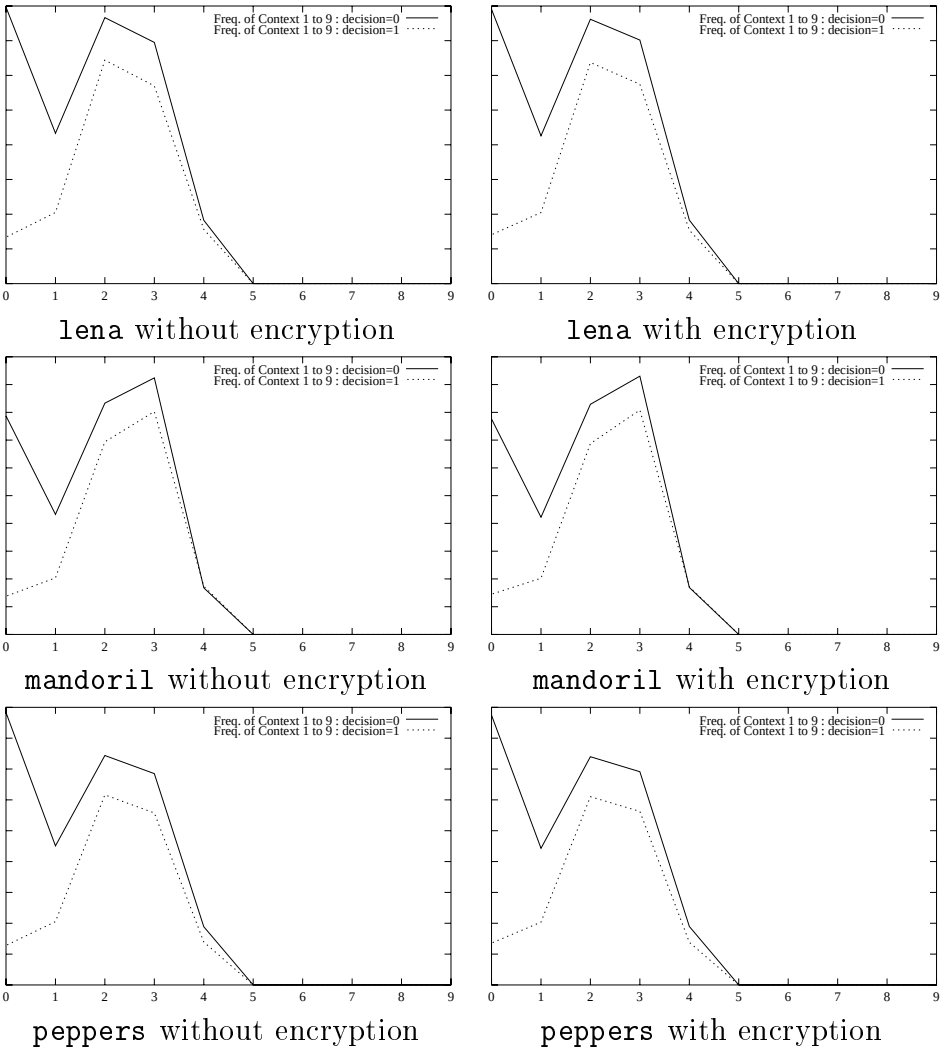


Figure 6.22: Frequencies of pairs of *contexts* and *decision* in the encoding of `lena.ppm`, `mandoril.ppm` and `peppers.ppm` without encryption (left column) and with encryption (right column).

Chapter 7

Image Authentication

7.1 Introduction

Image authentication aims to provide assurance about the integrity of images. In general, image data is in a compressed form due to its large size and the compression is lossy because of perceptual limitations of the human visual system. Because of this, unlike data authentication systems that must detect a single bit change in data, image authentication systems must remain tolerant to perceptually insignificant changes due to lossy compression.

JPEG [45] is the industry standard and is widely used in practice. In JPEG compression, by choosing different quality levels, the size of the output can be traded against the quality of the decompressed image. Using lower quality levels results in smaller file sizes at the expense of lower image quality after decompression.

An image authentication system that is tolerant to JPEG compression to a given quality level ℓ, must have the property that changes to the image resulting from compression to levels higher than ℓ, does not produce a 'false' response in the verification phase. However, malicious changes must be detectable. Compression tolerant image authentication systems can be broadly divided into *message authentication codes (MACs)* and *watermarking systems*. In the former, the aim of the system is to extract some *features* (also called signatures or image digests) of the image that remain invariant for images that have undergone JPEG compression to the given quality level. These features form *MACs* or *authentication tags* that will be appended to the image data and so an authenticated image is a pair consisting of the image and a tag.

In the latter approach, a watermark [124] signal is embedded into the image such that it can be recovered even if the image is compressed and decompressed. The advantage of the approach is that there is no need for a separate authenticator as the image carries the authenticating information with itself. However watermarking systems for

authentication must be fragile. That is, the watermark must be destroyed (become irrecoverable) with the slightest change to the image. However compression tolerance means that the watermark must survive changes that are due to JPEG compression algorithm. Reconciling these two requirements, that is fragility and compression tolerance is a challenge that must be addressed in this context. Another disadvantage of the watermarking approach is that the system embeds noise into an image and so it degrades the image quality. A number of systems from the latter type have been proposed but many of those based on fragile watermarking are less tolerant to JPEG compression [29]. Some of these systems have been shown to be insecure [75, 123], but many systems remain with no real security modeling or analysis.

In this chapter we will consider image authentication systems based on a MAC that are tolerant to the changes which are due to the JPEG image compression algorithm to a certain level compression quality. We review the JPEG compression system and a compression tolerant image authentication system called *SARI* by C. Lin and S. Chang [58] in Section 7.2. We also review an attack on this system proposed by Regunathan and Memon [82]. In Section 7.3 we present new attacks against SARI system and propose a method to improve security of the system. In Section 7.4 we propose a new compression tolerant image authentication system and finally we conclude.

7.2 Preliminaries

In this section first we review the JPEG image compression system and then following SARI image authentication system.

7.2.1 JPEG Compression

JPEG compression [45] is the image compression standard. JPEG, although it does have a lossless compression mode, is usually used as a lossy compression system and so the original and the compressed image have, in general, different values for the same pixel. In JPEG, the image is sub-divided into 8×8 pixel blocks. For each block, first the Discrete Cosine Transform (DCT) [3] coefficients are calculated, then quantized and then entropy-coded. The information loss is primarily due to quantization however computational error also contributes to the difference between the values of a pixel, before and after compression.

Let $P = \{p_1, p_2, \ldots, p_\wp\}$ denote the set of blocks, assuming that there are $\wp$ blocks in the image. For a real value R we write $R = h + r$ where h is an integer and

$r \in \mathbb{R}, -0.5 \leq r < 0.5$. Then the integer rounding function $rint()$ is defined as

$$\begin{aligned} rint(R) &= rint(h + r) \\ &= h \ . \end{aligned}$$

The main processing steps on a block during compression are the following.

- The DCT is applied to an 8×8 pixel block to produce 64 coefficients. The coefficients of a block p are written as ${F_p}^{(u,v)}$ $u, v \in \{1, 2, ..., 8\}$.
- Scalar quantization is used to obtain an integer value for ${F_p}^{(u,v)}$. Each coefficient is divided by an integer and the result is rounded. The quantization table is given by $Q^{(u,v)} \in \mathbb{N}$, $u, v \in \{1, 2, ..., 8\}$. The quantized value of the (u, v) coefficient in block p is given by

$$ {T_p}^{(u,v)} = rint(\frac{{F_p}^{(u,v)}}{Q^{(u,v)}}) \ . \tag{7.1}$$

- The quantized coefficients are entropy coded.

Decompression has the same three steps in reverse order. That is entropy decoding followed by dequantization of ${T_p}^{(u,v)}$, given by

$$\begin{aligned} \tilde{F}_p^{(u,v)} &= {T_p}^{(u,v)} Q^{(u,v)} \\ &= rint(\frac{{F_p}^{(u,v)}}{Q^{(u,v)}}) Q^{(u,v)} \end{aligned} \tag{7.2}$$

and finally applying the inverse DCT to reconstruct the image. The quality of the reconstructed image is determined by the *quality level*, which determines $Q^{(u,v)}$. The first and the last step of the compression are completely reversible (although in practice calculating DCT coefficients might introduce some computational error) but the second step is in general lossy and not reversible.

7.2.2 SARI Authentication System

Lin and Chang proposed a JPEG tolerant compression system [58], also known as the SARI system [57, 59]. The system operates on 8×8 Discrete Cosine Transformed image so it can be easily integrated into a JPEG compression system. The authors prove the soundness of the authentication system and argue that although it is possible to create tampered images that are acceptable by the authentication system, such images will include artifacts that make them detectable by human eyes. The system was later

shown to be insecure. In [82], Regunathan and Memon showed how to construct fraudulent images that are acceptable by SARI authentication system if the same key is used for signature generation of more than one image. However, the attack becomes ineffective if the signature is encrypted.

In Section 7.3 we show that with a relatively small amount of computation, it is possible to create a tampered image which is acceptable and the changes do not result in any visually detectable artifacts.

SARI system uses the property that the relative order of coefficients of a pair of blocks in the original image and the image after decompression remains the same. Hence the difference between two reconstructed coefficients can be bounded. That is, if $\Delta {F_{p,q}}^{(u,v)} = {F_p}^{(u,v)} - {F_q}^{(u,v)} > k$, then $\Delta \tilde{F}_{p,q}^{(u,v)} = \tilde{F}_p^{(u,v)} - \tilde{F}_q^{(u,v)}$ satisfies

$$\Delta \tilde{F}_{p,q}^{(u,v)} \geq \begin{cases} \tilde{k}^{(u,v)} Q^{(u,v)}, & \frac{k}{Q^{(u,v)}} \in \mathbb{Z} \\ (\tilde{k}^{(u,v)} - 1) Q^{(u,v)}, & \notin \mathbb{Z} \end{cases}$$

where $k \in \mathbb{R}$ is a fixed threshold and $\tilde{k}^{(u,v)} \equiv rint(\frac{k}{Q^{(u,v)}}), \forall u, v$.

Similarly, if ${\Delta F_{p,q}}^{(u,v)} < k$,

$$\Delta \tilde{F}_{p,q}^{(u,v)} \leq \begin{cases} \tilde{k}^{(u,v)} \cdot Q^{(u,v)}, & \frac{k}{Q^{(u,v)}} \in \mathbb{Z} \\ (\tilde{k}^{(u,v)} + 1) \cdot Q^{(u,v)}, & \frac{k}{Q^{(u,v)}} \notin \mathbb{Z} \end{cases}$$

and if ${\Delta F_{p,q}}^{(u,v)} = k$,

$$\Delta \tilde{F}_{p,q}^{(u,v)} = \begin{cases} \tilde{k}^{(u,v)} \cdot Q^{(u,v)}, & \frac{k}{Q^{(u,v)}} \in \mathbb{Z} \\ \tilde{k}^{(u,v)} \cdot Q^{(u,v)} \ or (\tilde{k}^{(u,v)} \pm 1) \cdot Q^{(u,v)}, & \frac{k}{Q^{(u,v)}} \notin \mathbb{Z} \end{cases}.$$

Generation of the signature

The signature is obtained by encoding the difference between coefficients of two blocks and generating a signature for all pairs of blocks and all the chosen frequencies.

A set selection algorithm is used to produce two sets of blocks, $P_p = \{p_1, p_2, \ldots, p_{\frac{\varrho}{2}}\}$ and $P_q = \{q_1, q_2, \ldots, q_{\frac{\varrho}{2}}\}$ that partition the set of image blocks, and then a pairing function is used to pair the blocks in the two sets. Finally, protected frequencies and the precisions of the frequencies are chosen.

The signature consists of all the *feature codes*, which are the encoded differences of coefficient pairs, together with the precision (number of bits) allocated to all frequencies.

Verification

The verification process uses the relationship between reconstructed coefficients in the two blocks. That is, if the difference between reconstructed coefficients is within an interval associated with the corresponding feature code and the pre-defined quantization error tolerance, the image is considered authentic. The interval is determined by the precision of the feature codes and acceptable quality level.

In this comparison, the effect of sources of error other than quantization, such as computational error due to the implementation of JPEG using finite precision arithmetic is also taken into account.

Evaluation

Security In this scheme, if the block pairing is public then a pair of blocks can be concurrently modified without the system detecting the modification. The authors of [57] argued that such an attack will result in noticeable artifacts and although undetectable by the verification system but will be visually detectable.

However, in general this is not true and methods such as those given in [52] can be used to modify a pair of blocks without creating any artifact. For security, it is necessary to hide block pairing but then the cost of finding the pairing is small. Our new attacks in Section 7.3.1 show that once the pairing is found, the attacker can simultaneously modify pairs of blocks and hence defeat the authentication system.

Length of the signature The size of the signature grows linearly with the number of frequencies that are protected. This number can range from 1 to 64. For each frequency different precision can be used. The length of the signature is

$$\frac{\wp}{2} \sum_{for\ all\ chosen\ (u,v)} b^{(u,v)}$$

where $b^{(u,v)}$ is the precision of frequency (u, v). For example, the size of the signature for a 512×512 image, protecting DC, AC1, AC2 and AC3 with 10 bits precision, is $(64 \times 64)/2 \times (10 + 10 + 10 + 10) = 81920$ bits (approximately 11kilo bytes).

7.3 New Attacks against the SARI System

In this section, we present new attacks against the SARI system [58]. We present ways of constructing fraudulent images that are accepted as authentic by the verification

system, and the modifications are visually undetectable. The attacks will work even if the feature code is encrypted. We also propose modifications to the system to make these attacks ineffective.

7.3.1 Attacks

Regunathan and Memon [82] showed a method of finding the secret block pairing if $O(log_{\wp})$ authenticated images using the same key are found. Once the pairing is discovered, the block pairs can be modified without being detected by the verification system. However an arbitrary modification, most likely, will result in visually detectable artifacts, as had been noted by the original authors [58]. Moreover, encrypting the feature code will make the attack completely ineffective.

We note that not all modifications result in visually detectable changes. For example, the method used in [52], uses combinations of DCT coefficients such that the central part of a block is modified in a smooth way, while the border of the block is unchanged. In Section 7.3.1, we describe an attack which generates a fraudulent image with no visual sign of being fraudulent, and succeeds even if the feature code is encrypted.

New Attacks

We consider two types of attacks.

1. The image is modified by simultaneously changing a pair of blocks by the same amount. The attack will succeed regardless of the precision of the feature codes and the number of protected coefficients. The modifications include,
   - adding or removing figures, letters and objects to the original image.
   - modifying figures, letters or objects in the original image.
2. If some of the coefficients are not protected, they can be arbitrarily changed.

Modifying Block Pairs

This attack succeeds if block pairing is known even if all the coefficients are protected and long precision feature codes are used. In Section 7.3.1 we show how to find the pairing even if the feature code is encrypted. The attack is made by modifying quantized coefficients of a JPEG compressed image by an equal amount in pairs of blocks.

Figures, letters or objects can be added or removed to pairs of blocks by adding an 8×8 block of quantized coefficients to the quantized coefficients of the pair. This is based on the following proposition.

Proposition 1 *Let D be an 8×8 pixel block, and $G(u,v)$, $u,v \in \{1,...,8\}$ denote its transformed and then quantized DCT coefficient in (u,v) position. Let $\tilde{F}_p(u,v)'$ and $\tilde{F}_q(u,v)'$ denote the coefficients of the reconstructed blocks corresponding to ${T_p}^{(u,v)} + G(u,v)$ and ${T_q}^{(u,v)} + G(u,v)$, respectively, and $\Delta\tilde{F}^{(u,v)}_{p,q}{}'$ denote the difference $\tilde{F}^{(u,v)}_{p}{}' - \tilde{F}^{(u,v)}_{q}{}'$. Then $\Delta\tilde{F}^{(u,v)}_{p,q} = \Delta\tilde{F}^{(u,v)}_{p,q}{}'$.*

Proof:

$$\begin{aligned}\Delta\tilde{F}^{(u,v)}_{p,q} &= {T_p}^{(u,v)} \cdot Q^{(u,v)} \\ &\quad -{T_q}^{(u,v)} \cdot Q^{(u,v)}\end{aligned}$$

and

$$\begin{aligned}\Delta\tilde{F}^{(u,v)}_{p,q}{}' &= {T_p}^{(u,v)}{}' \cdot Q^{(u,v)} \\ &\quad -{T_q}^{(u,v)}{}' \cdot Q^{(u,v)} \\ &= {T_p}^{(u,v)} \cdot Q^{(u,v)} \\ &\quad -{T_q}^{(u,v)} \cdot Q^{(u,v)} \\ &= \Delta\tilde{F}^{(u,v)}_{p,q} \ .\end{aligned}$$

This is true because of the following.

$$\begin{aligned}& {T_p}^{(u,v)}{}' \cdot Q^{(u,v)} - {T_q}^{(u,v)}{}' \cdot Q^{(u,v)} \\ = \ & ({T_p}^{(u,v)} + G(u,v)) \cdot Q^{(u,v)} \\ & \quad -({T_q}^{(u,v)} + G(u,v)) \cdot Q^{(u,v)} \\ = \ & {T_p}^{(u,v)} \cdot Q^{(u,v)} - {T_q}^{(u,v)} \cdot Q^{(u,v)} \ .\end{aligned}$$

■

Now suppose the attacker has a compressed image and its feature code. The attack is as follows.

1. The attacker creates the pattern D that is to be added to the block pair p and q.
2. He transforms D, and quantizes the coefficients by $Q^{(u,v)}$ to obtain $G^{(u,v)}$, $\forall u,v$.
3. Then he finds ${T_p}^{(u,v)} + G^{(u,v)}$ and ${T_q}^{(u,v)} + G^{(u,v)}$ and dequantizes the result.

We note that the system does not require the original image to verify the image and so the original image is not available for the verification.

The method will produce visually undetectable changes if the following conditions are satisfied.

C1 To make the block artifacts undetectable, the pixels on the edges of block D should be close to 0. This condition can be ignored if the pattern includes figures or letters with sharp edges at block edges.

C2 The difference of the modified block p and q must be exactly the same as the difference of the original p and q. Note that the use of the transformed version of the modified block without quantization by Q may result in the difference $\Delta\tilde{F}_{p,q}^{(u,v)}$ not to be a multiple of $Q^{(u,v)}$ although the original difference is a multiple of $Q^{(u,v)}$.

If the verification algorithm requires uncompressed image as input, we can generate it from the JPEG compressed image.

C3 G must be chosen such that the de-quantization and then the inverse-transform of the modified coefficients ${T_p}^{(u,v)} + G^{(u,v)}$ and ${T_q}^{(u,v)} + G^{(u,v)}$ do not produce a value outside the valid pixel range (for example, $[0, 255]$).

Figure 7.2 and 7.3 give an example of adding a pattern to the image. We followed the steps above, showing the case of pairing odd number and even number blocks for pattern "8" which is the pairing used in the original paper, and pairing distant location blocks for a pattern similar to "$\not\sim$".

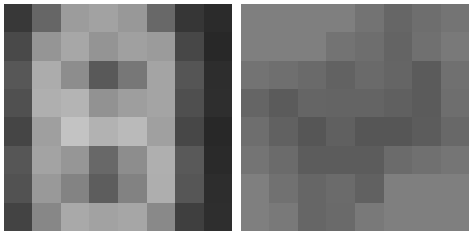


Figure 7.1: Pattern "8" (left) and a pattern similar to $\not\sim$ (right).

The pattern D that is added to the image is given in Figure 7.1. The attack succeeded because the two paired blocks have been smooth. We note that in the case of pattern "8", it is obvious that the same pattern appears twice but in the case of the other pattern, the modification on one of the two blocks (in the background) is not distinguishable. In some cases it might be more difficult to succeed. For example consider removing black numbers from a white license plate. Assume that the image

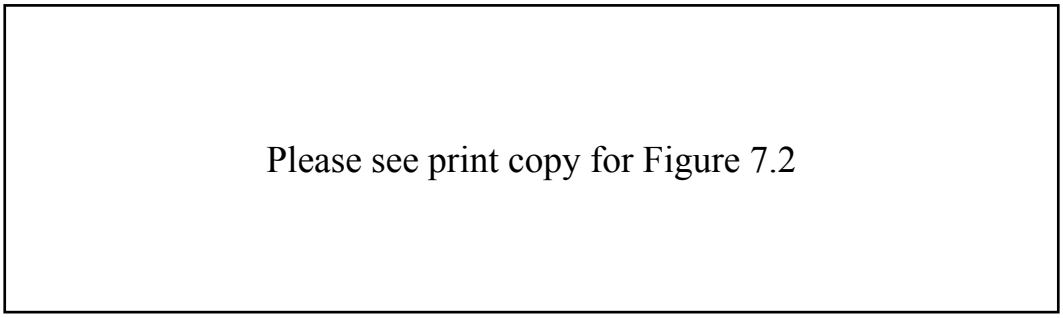


Figure 7.2: Example: Original image (left) and close up (right).

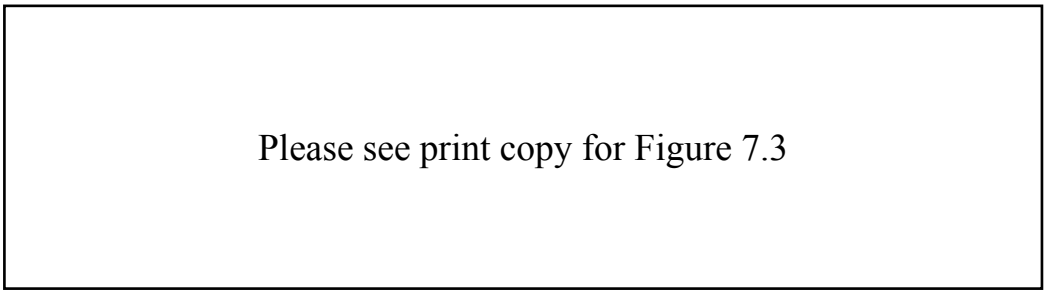


Figure 7.3: Close up of the modified image (left) and difference between the original and modified images (right). The large gray region, the darker part and the brighter part correspond to $\delta^{(i,j)} = 0$, $\delta^{(i,j)} < 0$ and $\delta^{(i,j)} > 0$, respectively.

is gray scale and pixel values are in the range $[0, 255]$, and the black and white pixel values are 0 and 255, respectively. Then to modify the numbers on the plate, some pixels need to be changed from black to white, or from white to black and so 255 has to be added to, or subtracted from these pixels, respectively. If the pixel to be modified is black and the pixel in the corresponding location in the paired block is white, then adding 255 to the pixel in the paired block will violate condition C3.

Let $r_p^{(i,j)}$,$i, j \in \{1, \cdots 8\}$, and $r_q^{(i,j)}$,$i, j \in \{1, \cdots 8\}$, denote the pixel values of block p and q, respectively, and $\delta^{(i,j)}$,$i, j \in \{1, \cdots 8\}$ denote the pixel values of the modification block.

Then the following must be satisfied.

$$\begin{aligned} 0 &\leq r_p^{(i,j)} + \delta^{(i,j)} \leq 255 \\ 0 &\leq r_q^{(i,j)} + \delta^{(i,j)} \leq 255 \end{aligned} \tag{7.3}$$

for all $i, j \in \{1, 2, ..., 8\}$.

For example, if we want $r_p^{(i,j)}$ to be as bright (i.e. large) as possible, we choose the largest possible $\delta^{(i,j)}$ that satisfies condition (7.3). That is, we choose $\min\{(255 - r_p^{(i,j)}), (255 - r_q^{(i,j)})\}$. If $r_q^{(i,j)}$ is large, then $255 - r_q^{(i,j)}$ is small and so is $\delta^{(i,j)}$. Hence,

$r_p^{(i,j)}$ cannot be increased by a large amount. From above, the range of $\delta^{(i,j)}$ is given as follows.

Theorem 2 *The range of* $\delta^{(i,j)}$ *is given by* $[0, \min\{(255 - r_p^{(i,j)}), (255 - r_q^{(i,j)})\}]$ *and* $[(-1)\min\{r_p^{(i,j)}, r_q^{(i,j)}\}, 0]$ *for the brightening and darkening modification, respectively.* ■

proof: For an 8 bit pixel $r_p^{(i,j)}$ and $r_q^{(i,j)}$, $0 \leq r_p^{(i,j)}$ and $r_q^{(i,j)} \leq 255$. From Condition (7.3), the possible minimum value of $\delta^{(i,j)}$ is 0 and the possible maximum value is either $255 - r_p^{(i,j)}$ or $255 - r_q^{(i,j)}$ and the smaller value of these two satisfies Condition (7.3).

Figure 7.5 shows the removal of letters from a license plate shown in Figure 7.4. Assuming even and odd block pairing, two horizontally neighboring blocks are modified. As an example, two digits were made bright so that it became the same color as the background of the plate.

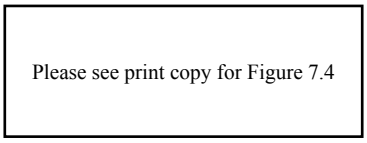


Figure 7.4: Original license plate.

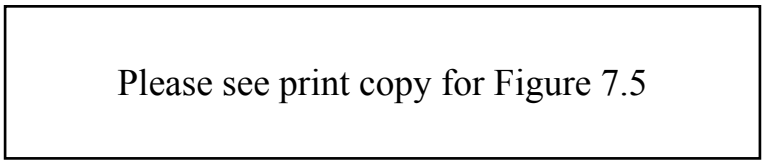


Figure 7.5: Removal experiments of "9" (left) and "5" (right).

From the above observations, we define a *vulnerable property* as follows.

Vulnerable property

If the range of $\delta^{(i,j)}$, given by Theorem 2, is large, then $r_p^{(i,j)}$ and $r_q^{(i,j)}$ are vulnerable to large modifications.

Finding Block-pairs

To increase security, block pairings can be kept secret. Suppose the attacker has an authenticated image (image together with its authenticator) and also access to a verification oracle: that is the verification program that inputs an image and its authenticator tag and produces a yes or no answer if the image does match or does not match the authenticator, respectively.

Algorithm 1 : finding a block pair.
1: The attacker chooses a block p_i to be modified.
2: loop until the pairing block is found.
3: Choose a block p_k, where $k \neq i$.
4: Modify p_i and p_k by the same amount.
5: Give the modified image to the oracle and observe its output.
6: If it is accepted
7: Exit the loop.

Note that the attacker does not have to find all block pairs but only those which he intends to modify.

The cost of finding p_k for a chosen p_i is $\wp - 1$. To find all block pairs, Algorithm 1 is iteratively applied to the blocks. In each iteration, it finds a pair. Initially there are $\wp/2$ pairs to find and in the first iteration it tries at most $\wp - 1$ blocks. Then the number of pairs becomes $\wp/2 - 1$ in the second iteration and it experiments $\wp - 3$ blocks. The number of blocks to be examined at ith iteration is $\wp - (2i - 1)$. There are $\wp/2$ pairs and so the cost of finding all pairs is

$$\begin{aligned} & \sum_{i=1}^{\wp/2} (\wp - (2i-1)) \\ = \; & \wp^2/2 - \wp^2/4 \\ = \; & \wp^2/4 \; . \end{aligned}$$

For example, the 512×512 image `lena` has 4096 blocks. The cost of finding p_k for a chosen p_i is $4095 \approx 2^{12}$ and that of finding all pairs is 2^{22}, which is considered small in cryptographic systems. If each of 64 frequencies uses a different pairing, each pairing can be independently found and in this case, the cost of finding a single and all pairs becomes $2^{12} \times 64 = 2^{18}$, and $2^{22} \times 64 = 2^{28}$, respectively.

Attack on Unprotected Coefficients

When only some of the coefficients are protected, the unprotected ones can be arbitrarily modified. Because of visual significance of lower frequencies, it is more likely that they will be chosen for protection. So if the added pattern is obtained by modifying the higher frequency components, then the resulting modification will look like spraying the image with black or white dots.

Figure 7.6 shows an example of such attacks.

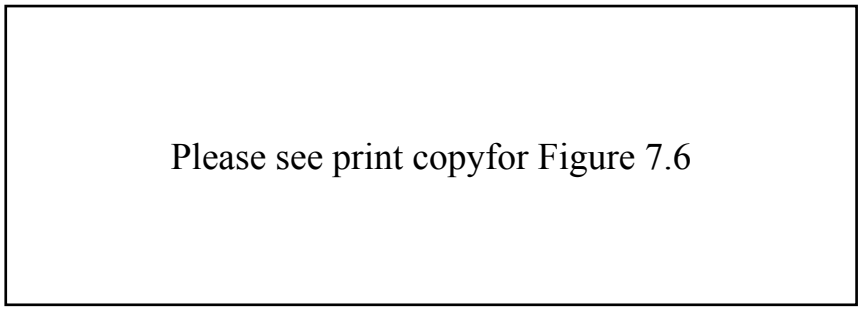


Figure 7.6: The two images will be authenticated with the coefficients 0-10 (left) and 0-59 (right) protected.

7.3.2 Improvement

The attacks in [82] and our new attacks clearly show that simply hiding the block-pairing will not add security because each signature bit can be tied to a single pair and so the pairing can be found easily. If we allow the pairs to overlap, that is, allow a block to be shared by more than one pair, then a signature bit will be linked to more than a single pair.

Let a subset of $\frac{\wp}{2}$ pairs be $\mathcal{S}_i$ consisting of s pairs and including $s+1$ blocks, such that every pair in $\mathcal{S}_i$ has a common block with one other element of $\mathcal{S}_i$. Assuming $\mathcal{S}_i \cap \mathcal{S}_j = \phi$, $i \neq j$, the number of $\mathcal{S}_i$ required to include $\wp$ blocks is $\frac{\wp}{s+1}$ and the number of pairs in $\mathcal{S}_i$,$\forall i$ is $\frac{\wp s}{s+1}$.

For example if two pairs share a block, the computational cost of the attack will increase. Let $\mathcal{S}_i = \{(p_{a1}, p_{a2}), (p_{a2}, p_{a3})\}$. Assuming an attacker tries to find a block paired to p_{a1} among $\wp - 1$ blocks, he needs to modify Algorithm 1 in Section 7.3.1 such that he modifies a triplet of blocks by the same amount instead of a block pair. This increases the order of the cost to $O(\wp^3)$ from $O(\wp^2)$. In general the cost of finding $s+1$ blocks in $\mathcal{S}_i$ will be of the order $O(\wp^{s+1})$.

The disadvantage of this method is that the number of pairs increases $\frac{2s}{s+1} = 2 - \frac{2}{s+1}$ times compared with the original system, and so the signature size increases. For example, if $\mathcal{S}_i$ consists of two pairs sharing one block, i.e. $s = 2$, the signature size is $\frac{4}{3}$ times larger than the signature generated by the original system. To reduce the signature size, the method can be applied to a selected set of blocks. We suggest the following two approaches.

Approach 1

1. First construct pairs so that they do not have *vulnerable property* given in Section 7.3.1.

2. If there are pairs which have *vulnerable property*, then reconstruct pairs for these

blocks using the method of sharing blocks in pairs.

Approach 2

The important blocks are interactively selected, i.e. a user chooses their *region of interest* interactively when the signature is generated. Then the above method is applied to these chosen blocks.

7.3.3 Concluding Remarks

We have shown methods of modifying authenticated images, which are visually undetectable and pass the verification test, and defined the vulnerable property of pixels. Although modifications are restricted on pixels which have the vulnerable property, if the system fails to provide the assurance of protecting these pixels, then images are vulnerable against the attack. We showed a modification to the system which increases the cost of the attack to $O(\wp^{s+1})$ to the extent that the system can be considered secure.

7.4 A Secure and Flexible Authentication System for Digital Images

In this section we propose an image authentication system in which the MAC remains invariant after JPEG compression to a given quality level. We propose a model for analyzing security which captures the attacking power of a typical adversary in a real-life application of the system. To our knowledge this is the first clear model for security evaluation of image authentication systems. We show that the proposed system is secure in this model and verify this result by a number of experiments.

The system has a number of attractive properties.

1. The main computation of the system is in computing the DCT of image blocks which is part of the JPEG compression algorithm. This means that the system can be effectively integrated into the compression system. This is particularly important because in many applications image processing hardware is not suitable for traditional cryptographic operations and so calculating cryptographic checksums either requires a cryptographic co-processor, or will significantly slow down the system.

2. The computation is parallelizable and the system can be used for real-time data. The system is stream oriented. That is, as data arrives it is processed and

the checksum is generated accordingly. The system can be easily extended to frame-based moving picture data (MJPEG [102]) and with more efforts to motion compensated compression systems, such as MPEG (Moving Pictures Experts Group [42]).

3. The system has flexible protection. That is, by allowing longer checksums the level of protection can be increased. The system allows selective protection: that is the key information may be chosen in an image dependent way such that the sensitive parts of the image receive higher protection. This is a very useful property that can be be used to protect regions of interest in the image.

In Section 7.4.1 we describe our system and show its properties. Section 7.4.2 gives the design of our system and in Section 7.4.4 we analyze the security of our system and finally we conclude.

7.4.1 A Secure and Flexible Authentication Scheme

We propose a message authentication code (MAC) that consists of *feature codes obtained by encoding a linear combination of DCT coefficients in subsets of blocks.* A MAC is generated from an original image and it corresponds to all images which are created by compressing the original image using JPEG compression with various quality levels. The image is partitioned into subsets of blocks, called *groups* and blocks in each group are used to generate feature codes. This system can be considered as a generalization of the SARI system.

In the following, without loss of generality, we assume ${A_i}^{(u,v)}$ are non-negative integer constants. The approach can be used for any value of ${A_i}^{(u,v)}$.

Let $\{G_1, G_2 \cdots\}$ be a set of $\frac{\wp}{m}$ groups, each group consisting of m blocks, such that $\cup_j G_j = P$, and $G_j \cap G_h = \phi$ for all i and j.

The outline of the MAC generation algorithm is as follows.

1. For all blocks in P, obtain the 64 DCT coefficients.

2. Let ${F_{i,j}}^{(u,v)}$ denote the DCT coefficient in (u, v) position of the ith block in G_j. Then ${Y_j}^{(u,v)} = \sum_{\forall i \in [m]} {A_i}^{(u,v)} {F_{i,j}}^{(u,v)}$ is the weighted sum of all coefficients in ${G_j}^{(u,v)}$.

3. The feature code is generated by encoding ${Y_j}^{(u,v)}$ as shown in this section.

Theorem 3 shows that the same linear combination in the reconstructed image is closely related to that of the original image. This property forms the basis of correct verification.

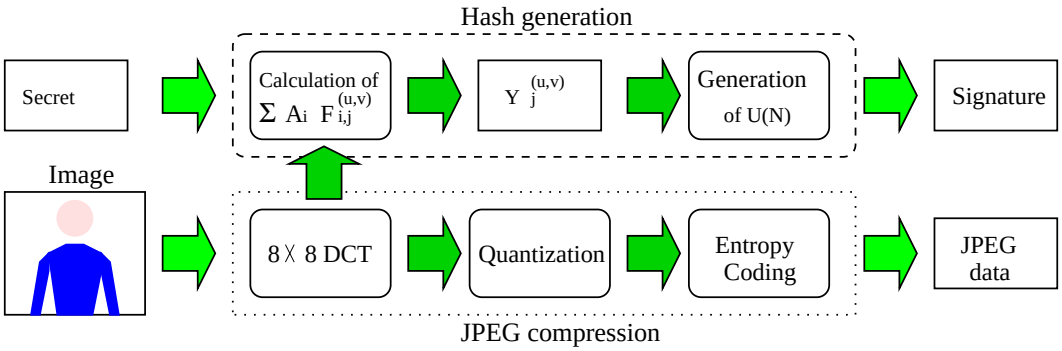


Figure 7.7: MAC generation and JPEG compression.

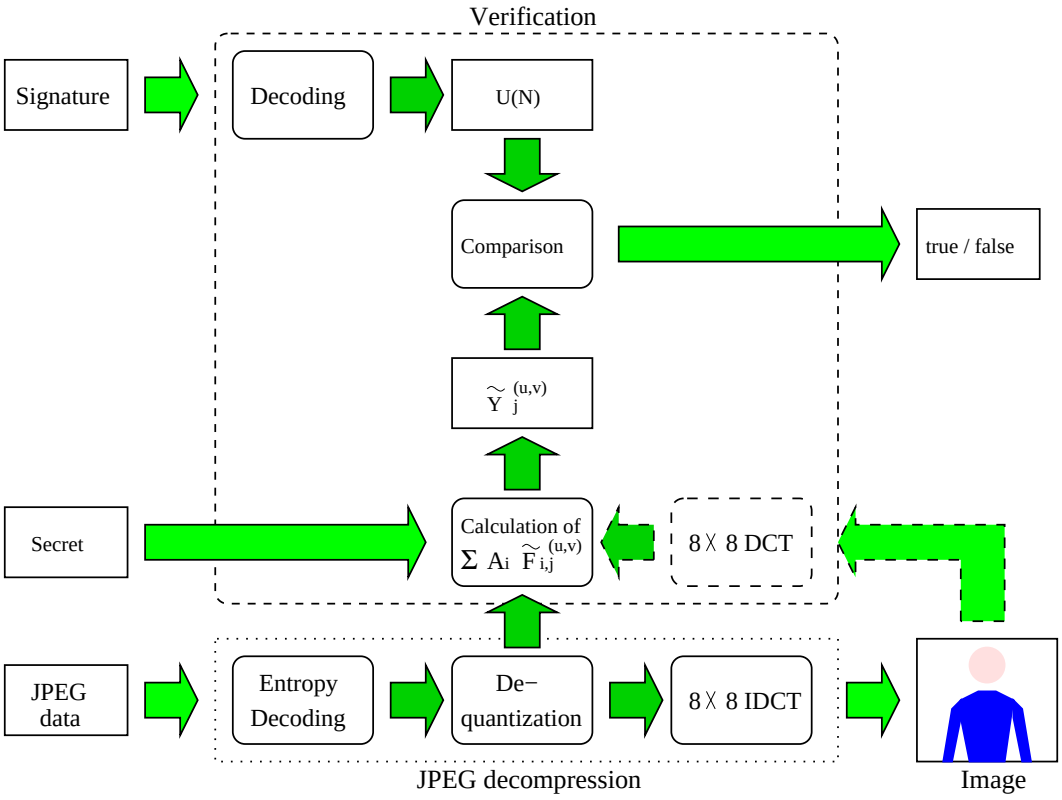


Figure 7.8: MAC verification and JPEG decompression.

Let $i \in \{1, 2, 3, ..., m\}$.

Let $\frac{F_p^{(u,v)}}{Q^{(u,v)}}$ be $\frac{F_p^{(u,v)}}{Q^{(u,v)}} = R_p = h_p + r_p$ where h_p is an integer and $-0.5 \leq r_p < 0.5$. Then, $F_p^{(u,v)}$ and $\tilde{F}_p^{(u,v)}$ are the original and reconstructed values of a DCT coefficient.

$$F_p^{(u,v)} = R_p Q^{(u,v)}$$

$$= h_p Q^{(u,v)} + r_p Q^{(u,v)}$$

$$\begin{aligned} \tilde{F}_p^{(u,v)} &= rint(R_p)Q^{(u,v)} \\ &= h_p Q^{(u,v)} \\ &= {F_p}^{(u,v)} - r_p Q^{(u,v)} \ . \end{aligned} \tag{7.4}$$

As noted in [82] and Section 7.2.2 using a pair of blocks allows the attacker to find the pairing and find two images with the same MAC value. Using the combination of many blocks effectively links a large number of blocks together and makes it much more difficult for the attacker to determine the groups that are linked. The weighting can be used to emphasize regions of interest in the image.

In the following we prove a theorem which shows that the reconstructed value of ${Y_j}^{(u,v)}$ can be bounded. This property can be used to estimate the difference between the linear sum in the reconstructed image and that in the original image and so can be used to detect tampering with the image. (For simplicity and without loss of generality, we omit the floor and ceiling in the following.)

Theorem 3 *Let* $\tilde{K} = \tilde{k}^{(u,v)}Q^{(u,v)}$, ${Y_j}^{(u,v)}$ *as defined above,* $\tilde{Y}_j^{(u,v)} = \sum_{\forall i \in [m]} {A_i}^{(u,v)} \tilde{F}_{i,j}^{(u,v)}$, *and* $L = \sum_{\forall i \in [m]} |{A_i}^{(u,v)}|$.

Then for all j, *the relationship between* ${Y_j}^{(u,v)}$ *and* $\tilde{Y}_j^{(u,v)}$ *is given as follows:*

1. *If* ${Y_j}^{(u,v)} = k$,

$$\begin{aligned} (\tilde{k}^{(u,v)} - 0.5(1 + \sum_{\forall i \in [m]} |{A_i}^{(u,v)}|))Q^{(u,v)} &< \tilde{Y}_j^{(u,v)} \\ &< (\tilde{k}^{(u,v)} + 0.5(1 + \sum_{\forall i \in [m]} |{A_i}^{(u,v)}|))Q^{(u,v)} \ . \end{aligned}$$

2. *If* ${Y_j}^{(u,v)} < k$, *then*

$$\tilde{Y}_j^{(u,v)} < \tilde{k}^{(u,v)}Q^{(u,v)} + 0.5Q^{(u,v)}(1 + \sum_{\forall i \in [m]} |{A_i}^{(u,v)}|) \ .$$

3. *If* ${Y_j}^{(u,v)} > k$, *then*

$$\tilde{Y}_j^{(u,v)} > \tilde{k}^{(u,v)}Q^{(u,v)} - 0.5Q^{(u,v)}(1 + \sum_{\forall i \in [m]} |{A_i}^{(u,v)}|) \ . \tag{7.5}$$

Proof of Theorem 3:

We have

$$k - 0.5Q^{(u,v)} \leq \tilde{K} < k + 0.5Q^{(u,v)}$$

and from Theorem 4

$$Y_j{}^{(u,v)} - 0.5Q^{(u,v)}L < \tilde{Y}_j^{(u,v)} \leq Y_j{}^{(u,v)} + 0.5Q^{(u,v)}L$$

The relationship between $Y_j{}^{(u,v)} - k$ and $\tilde{Y}_j^{(u,v)} - \tilde{k}^{(u,v)}Q^{(u,v)}$ is given by

$$Y_j{}^{(u,v)} - 0.5Q^{(u,v)}L - (k + 0.5Q^{(u,v)}) < \tilde{Y}_j^{(u,v)} - \tilde{K} \leq Y_j{}^{(u,v)} + 0.5Q^{(u,v)}L - (k - 0.5Q^{(u,v)})$$

and so

$$(Y_j{}^{(u,v)} - k) - 0.5Q^{(u,v)}(L+1) < \tilde{Y}_j^{(u,v)} - \tilde{K} \leq (Y_j{}^{(u,v)} - k) + 0.5Q^{(u,v)}(L+1) \ . \quad (7.6)$$

Now we have the following cases.

1. If $Y_j{}^{(u,v)} = k$, and so $Y_j{}^{(u,v)} - k = 0$ in equation (7.6) and so

$$-0.5Q^{(u,v)}(L+1) < \tilde{Y}_j^{(u,v)} - \tilde{K} \leq 0.5Q^{(u,v)}(L+1)$$

That is

$$\tilde{K} - 0.5Q^{(u,v)}(L+1) < \tilde{Y}_j^{(u,v)} \leq \tilde{K} + 0.5Q^{(u,v)}(L+1)$$

and

$$\begin{aligned} \tilde{k}^{(u,v)}Q^{(u,v)} - 0.5Q^{(u,v)}(1 + \sum_{\forall i \in [m]} |A_i{}^{(u,v)}|) < \tilde{Y}_j^{(u,v)} \\ \leq \tilde{k}^{(u,v)}Q^{(u,v)} + 0.5Q^{(u,v)}(1 + \sum_{\forall i \in [m]} |A_i{}^{(u,v)}|) \ . \end{aligned}$$

Note that because $A_i{}^{(u,v)}$ and $\tilde{F}_{i,j}^{(u,v)}$ are integers, the sum $\sum_{\forall i \in [m]} A_i{}^{(u,v)}\tilde{F}_{i,j}^{(u,v)}$ is an integer and so the above relationship becomes

$$\begin{aligned} \lceil \tilde{k}^{(u,v)}Q^{(u,v)} - 0.5Q^{(u,v)}(1 + \sum_{\forall i \in [m]} |A_i{}^{(u,v)}|) \rceil \leq \tilde{Y}_j^{(u,v)} \\ \leq \lfloor \tilde{k}^{(u,v)}Q^{(u,v)} + 0.5Q^{(u,v)}(1 + \sum_{\forall i \in [m]} |A_i{}^{(u,v)}|) \rfloor \ . \end{aligned}$$

2. If $Y_j{}^{(u,v)} < k$, and so $Y_j{}^{(u,v)} - k < 0$ in equation (7.6) and $(Y_j{}^{(u,v)} - k) + 0.5Q^{(u,v)}(L+1) < 0.5Q^{(u,v)}(L+1)$. Then $\tilde{Y}_j^{(u,v)} - \tilde{K}$ always satisfies

$$\tilde{Y}_j^{(u,v)} - \tilde{K} < 0.5Q^{(u,v)}(L+1)$$

and so

$$\tilde{Y}_j^{(u,v)} < \tilde{K} + 0.5Q^{(u,v)}(L+1) \ .$$

That is,

$$\tilde{Y}_j^{(u,v)} < \tilde{k}^{(u,v)}Q^{(u,v)} + 0.5Q^{(u,v)}(1 + \sum_{\forall i \in [m]} |A_i{}^{(u,v)}|) \ .$$

Note that the sum $\tilde{Y}_j^{(u,v)}$ is an integer because $A_i{}^{(u,v)}$ and $\tilde{F}_{i,j}^{(u,v)}$ are integers and so we have

$$\tilde{Y}_j^{(u,v)} \leq \lfloor \tilde{k}^{(u,v)}Q^{(u,v)} + 0.5Q^{(u,v)}(1 + \sum_{\forall i \in [m]} |A_i{}^{(u,v)}|) \rfloor \ .$$

3. If $Y_j{}^{(u,v)} > k$, then $Y_j{}^{(u,v)} - k > 0$ in equation (7.6) and so as k approaches S, $Y_j{}^{(u,v)} - k$ approaches 0 and $(Y_j{}^{(u,v)} - k) - 0.5Q^{(u,v)}(L+1)$ approaches $-0.5Q^{(u,v)}(L+1)$. Since $Y_j{}^{(u,v)} - k > 0$, $(Y_j{}^{(u,v)} - k) - 0.5Q^{(u,v)}(L+1) > -0.5Q^{(u,v)}(L+1)$. Then $\tilde{Y}_j^{(u,v)} - \tilde{K}$ satisfies

$$\tilde{Y}_j^{(u,v)} - \tilde{K} > -0.5Q^{(u,v)}(L+1)$$

and so

$$\tilde{Y}_j^{(u,v)} > \tilde{K} - 0.5Q^{(u,v)}(L+1) \ .$$

That is,

$$\tilde{Y}_j^{(u,v)} > \tilde{k}^{(u,v)}Q^{(u,v)} - 0.5Q^{(u,v)}(1 + \sum_{\forall i \in [m]} |A_i{}^{(u,v)}|) \ .$$

Note that the sum $\tilde{Y}_j^{(u,v)}$ is an integer because $A_i{}^{(u,v)}$ and $\tilde{F}_{i,j}^{(u,v)}$ are integers and so the above relationship becomes

$$\tilde{Y}_j^{(u,v)} \geq \lceil \tilde{k}^{(u,v)}Q^{(u,v)} - 0.5Q^{(u,v)}(1 + \sum_{\forall i \in [m]} |A_i{}^{(u,v)}|) \rceil \ .$$

Let $A_i{}^{(u,v)}$ be integers. Then the relationship between $\sum_{\forall i \in [m]} A_i{}^{(u,v)} F_i^{(u,v)}$ and $\sum_{\forall i \in [m]} A_i{}^{(u,v)} \tilde{F}_i^{(u,v)}$ is as follows.

Theorem 4 *The sum $\sum_{\forall i \in [m]} A_i{}^{(u,v)} \tilde{F}_i^{(u,v)}$ is bounded as follows. (The difference in the three cases is including or excluding equality).*

- *if* $A_i{}^{(u,v)} \geq 0 \ \forall i$,

$$\sum_{\forall i \in [m]} A_i{}^{(u,v)} F_i{}^{(u,v)} - 0.5Q^{(u,v)} \sum_{\forall i \in [m]} |A_i{}^{(u,v)}| < \sum_{\forall i \in [m]} A_i{}^{(u,v)} \tilde{F}_i^{(u,v)} \leq \sum_{\forall i \in [m]} A_i{}^{(u,v)} F_i{}^{(u,v)} + 0.5Q^{(u,v)} \sum_{\forall i \in [m]} |A_i{}^{(u,v)}| \ .$$

- *if* $A_i{}^{(u,v)} < 0 \ \forall i$,

$$\sum_{\forall i \in [m]} A_i{}^{(u,v)} F_i{}^{(u,v)} - 0.5Q^{(u,v)} \sum_{\forall i \in [m]} |A_i{}^{(u,v)}| \leq \sum_{\forall i \in [m]} A_i{}^{(u,v)} \tilde{F}_i^{(u,v)} < \sum_{\forall i \in [m]} A_i{}^{(u,v)} F_i{}^{(u,v)} + 0.5Q^{(u,v)} \sum_{\forall i \in [m]} |A_i{}^{(u,v)}| \ .$$

- *if* $A_i{}^{(u,v)}$ *includes negative and positive integers,*

$$\sum_{\forall i \in [m]} A_i{}^{(u,v)} F_i{}^{(u,v)} - 0.5Q^{(u,v)} \sum_{\forall i \in [m]} |A_i{}^{(u,v)}| < \sum_{\forall i \in [m]} A_i{}^{(u,v)} \tilde{F}_i^{(u,v)} < \sum_{\forall i \in [m]} A_i{}^{(u,v)} F_i{}^{(u,v)} + 0.5Q^{(u,v)} \sum_{\forall i \in [m]} |A_i{}^{(u,v)}| \ .$$

Proof:

Since $-0.5 \leq r_p < 0.5$,

$$F_p{}^{(u,v)} - 0.5Q^{(u,v)} < F_p{}^{(u,v)} - r_p Q^{(u,v)} \leq F_p{}^{(u,v)} + 0.5Q^{(u,v)} \tag{7.7}$$

and so

$$F_p{}^{(u,v)} - 0.5Q^{(u,v)} < \tilde{F}_p^{(u,v)} \leq F_p{}^{(u,v)} + 0.5Q^{(u,v)} \ . \tag{7.8}$$

We have

$$\begin{aligned} \sum_{\forall i \in [m]} A_i{}^{(u,v)} F_i{}^{(u,v)} &= Q^{(u,v)} \sum_{\forall i \in [m]} A_i{}^{(u,v)} h_i + Q^{(u,v)} \sum_{\forall i \in [m]} A_i{}^{(u,v)} r_i \\ &= Q^{(u,v)} \sum_{\forall i \in [m]} A_i{}^{(u,v)} \tilde{F}_i^{(u,v)} + Q^{(u,v)} \sum_{\forall i \in [m]} A_i{}^{(u,v)} r_i \end{aligned}$$

and

$$\begin{aligned} \sum_{\forall i \in [m]} A_i{}^{(u,v)} \tilde{F}_i^{(u,v)} &= Q^{(u,v)} \sum_{\forall i \in [m]} A_i{}^{(u,v)} h_i \\ &= \sum_{\forall i \in [m]} A_i{}^{(u,v)} F_i{}^{(u,v)} - Q^{(u,v)} \sum_{\forall i \in [m]} A_i{}^{(u,v)} r_i \ . \end{aligned}$$

Since $-0.5 \leq r_i < 0.5$,

- if ${A_i}^{(u,v)} \geq 0$, i.e. ${A_i}^{(u,v)} = |{A_i}^{(u,v)}|$,

$$-0.5|{A_i}^{(u,v)}|Q^{(u,v)} \leq {A_i}^{(u,v)}Q^{(u,v)}r_i < 0.5|{A_i}^{(u,v)}|Q^{(u,v)}$$

and from equation (7.8),

$${A_i}^{(u,v)}F_i^{(u,v)} - 0.5|{A_i}^{(u,v)}|Q^{(u,v)} < {A_i}^{(u,v)}\tilde{F}_i^{(u,v)} \leq {A_i}^{(u,v)}F_i^{(u,v)} + 0.5|{A_i}^{(u,v)}|Q^{(u,v)} .$$

- if ${A_i}^{(u,v)} < 0$, i.e. ${A_i}^{(u,v)} = -|{A_i}^{(u,v)}|$,

$$-0.5|{A_i}^{(u,v)}|Q^{(u,v)} < {A_i}^{(u,v)}Q^{(u,v)}r_i \leq 0.5|{A_i}^{(u,v)}|Q^{(u,v)}$$

and from equation (7.8),

$${A_i}^{(u,v)}F_i^{(u,v)} - 0.5|{A_i}^{(u,v)}|Q^{(u,v)} \leq {A_i}^{(u,v)}\tilde{F}_i^{(u,v)} < {A_i}^{(u,v)}F_i^{(u,v)} + 0.5|{A_i}^{(u,v)}|Q^{(u,v)} \quad (7.9)$$

The theorem follows by summing equation (7.9) for all $i \in [m]$.

Feature Code

A feature code is a binary string representing ${Y_j}^{(u,v)}$. The coding process also generates an interval whose length determines the acceptable accuracy. For verification, an error tolerance value is needed. This can be chosen by considering the acceptable quality levels of the compression algorithm.

Encoding starts with an initial interval. This interval is then halved and labeled by 0, for the lower, and 1 for the upper subintervals. The first bit of feature code is obtained by determining the subinterval that ${Y_j}^{(u,v)}$ belongs to. The initial interval is now replaced by the subinterval containing ${Y_j}^{(u,v)}$. In each step a bit is generated by determining the subinterval containing ${Y_j}^{(u,v)}$ and the step is repeated.

Let $[{F_{MIN}}^{(u,v)}, {F_{MAX}}^{(u,v)})$ be the range of the DCT coefficients in the (u,v) position, and $Z_{j,1}^{(u,v)}, Z_{j,2}^{(u,v)}, ..., Z_{j,N}^{(u,v)}$ denote the bit sequence generated from ${Y_j}^{(u,v)}$. For example, ${F_{MIN}}^{(u,v)} = 0$ and ${F_{MAX}}^{(u,v)} = 2048$ for DC coefficients and ${F_{MIN}}^{(u,v)} = -1024$ and ${F_{MAX}}^{(u,v)} = 1024$ for AC coefficients. Let $I(n)$ and $U(n)$ be defined as follows.

$$I(n) = ({F_{MAX}}^{(u,v)} - {F_{MIN}}^{(u,v)})2^{-n}, n \in \{1, 2, 3, ..., N\} \quad (7.10)$$

and for all $n \in \{1, 2, 3, ..., N\}$

$$\begin{aligned} U(0) &= {F_{MIN}}^{(u,v)} \\ U(n) &= U(n-1) + Z_{j,n}^{(u,v)} I(n) \end{aligned}$$

$$= \quad {F_{MIN}}^{(u,v)} + ({F_{MAX}}^{(u,v)} - {F_{MIN}}^{(u,v)}) \sum_{l=1}^{N} Z_{j,l}^{(u,v)} 2^{-l} \ . \tag{7.11}$$

The coding procedure for ${Y_j}^{(u,v)}$ is given in Algorithm 2.

	Algorithm 2 : Coding ${Y_j}^{(u,v)}$
1:	Initially the interval d is $d = [U(0), U(0) + 2I(1)) = [{F_{MIN}}^{(u,v)}, {F_{MAX}}^{(u,v)})$.
2:	$n = 0$.
3:	Repeat while $(n \leq N)$
4:	$n = n + 1$.
5:	Divide $d = [U(n-1), U(n-1) + 2I(n))$ into two intervals,
	$d_l = [U(n-1), U(n-1) + I(n))$
	and $d_u = [U(n-1) + I(n), U(n-1) + 2I(n))$.
6:	if ${Y_j}^{(u,v)} \in d_u$,
7:	$Z_{j,1}^{(u,v)} = 1$
8:	else if ${Y_j}^{(u,v)} \in d_l$,
9:	$Z_{j,1}^{(u,v)} = 0$
10:	output $Z_{j,1}^{(u,v)}$
11:	$d = [U(n-1) + Z_{j,1}^{(u,v)} I(n), U(n-1) + I(n) + Z_{j,1}^{(u,v)} I(n))$
	$= [U(n), U(n) + 2I(n+1))$

After n rounds, $[U(n), U(n) + 2I(n+1))$ is the interval containing ${Y_j}^{(u,v)}$.

The feature code generated above gives a binary representation for ${Y_j}^{(u,v)}$ and $U(N)$ gives the precision interval for ${Y_j}^{(u,v)}$. That is,

$$U(N) \leq {Y_j}^{(u,v)} < U(N) + ({F_{MAX}}^{(u,v)} - {F_{MIN}}^{(u,v)}) 2^{-N} \ . \tag{7.12}$$

Finding the Tolerance Interval

The difference between ${Y_j}^{(u,v)}$ and the decompressed value $\tilde{Y}_j^{(u,v)}$ is due to quantization error and calculation errors. In the following we find an estimate of the two types of errors. Using these two estimates we can determine an error tolerance interval corresponding to acceptable compression quality level.

Quantization Error

The quantization error $\Delta = \tilde{Y}_j^{(u,v)} - {Y_j}^{(u,v)}$ is the sum of m random variables, each corresponding to the quantization error of a single coefficient. That is,

$$\Delta = \sum_{i=1}^{m} \delta_i \text{ where } \delta_i = \tilde{F}_{i,j}^{(u,v)} - {F_{i,j}}^{(u,v)} \ .$$

To model the behavior of this variable, we have conducted a number of experiments reported in Section 7.4.6. Our experimental results on the distribution of the quantization error of a single coefficient (linear combination of size one) is in Section 7.4.6 and shows that *i*) at lower quality levels and lower frequencies, the distribution of the quantization error values is close to the uniform distribution and *ii*) at higher quality levels and higher frequencies, it is a symmetric Gaussian-like distribution with zero-mean. In between the two, that is for other quality levels and other frequencies the distribution is also Gaussian-like always with zero mean and the variance depending on the quality level.

An interesting observation is that when m is large, the sum of the quantization errors will have a smaller variance. This is expected because assuming each error δ_i is normally distributed and has variance ${\sigma_i}^2$, then Δ will have the variance $\frac{\sum {\sigma_i}^2}{m}$. This means that $\tilde{Y}_j^{(u,v)}$ will be closer to ${Y_j}^{(u,v)}$ and hence a tighter interval for $\tilde{Y}_j^{(u,v)}$ is resulted.

Computation Error

Let $\varepsilon_i^{(u,v)} \in \mathbb{R}$ denote the computation error in calculating $\tilde{F}_{i,j}^{(u,v)}$ due to the finite precision calculations used in the implementation of JPEG and other sources such as integer representation of real value coefficients. Computation errors introduce inaccuracy in $\tilde{F}_{i,j}^{(u,v)}$, that is $\tilde{F}_{i,j}^{(u,v)} = rint(\frac{{F_{i,j}}^{(u,v)} + \varepsilon_i^{(u,v)}}{Q^{(u,v)}})Q^{(u,v)}$. Then equation (7.9) becomes

$$\begin{aligned} {A_i}^{(u,v)}({F_{i,j}}^{(u,v)} + \varepsilon_i^{(u,v)}) - 0.5|{A_i}^{(u,v)}|Q^{(u,v)} &\leq {A_i}^{(u,v)}\tilde{F}_{i,j}^{(u,v)} \\ &< {A_i}^{(u,v)}({F_{i,j}}^{(u,v)} + \varepsilon_i^{(u,v)}) + 0.5|{A_i}^{(u,v)}|Q^{(u,v)} \end{aligned}$$

and Theorem 4 becomes

$$\begin{aligned} \sum_{\forall i \in [m]} {A_i}^{(u,v)}{F_{i,j}}^{(u,v)} - 0.5Q^{(u,v)} \sum_{\forall i \in [m]} |{A_i}^{(u,v)}| + \sum_{\forall i \in [m]} {A_i}^{(u,v)}\varepsilon_i^{(u,v)} &< \sum_{\forall i \in [m]} {A_i}^{(u,v)}\tilde{F}_{i,j}^{(u,v)} \\ \leq \sum_{\forall i \in [m]} {A_i}^{(u,v)}{F_{i,j}}^{(u,v)} + 0.5Q^{(u,v)} \sum_{\forall i \in [m]} |{A_i}^{(u,v)}| + \sum_{\forall i \in [m]} {A_i}^{(u,v)}\varepsilon_i^{(u,v)} &. \end{aligned}$$

We can ignore the error in calculation of $\tilde{k}^{(u,v)}$, by using high enough precision and so Theorem 3 becomes

1. If ${Y_j}^{(u,v)} = k$,

$$\begin{aligned} (\tilde{k}^{(u,v)} - 0.5(1 + \sum_{\forall i \in [m]} |{A_i}^{(u,v)}|))Q^{(u,v)} + \sum_{\forall i \in [m]} {A_i}^{(u,v)}\varepsilon_i^{(u,v)} &< \tilde{Y}_j^{(u,v)} \\ < (\tilde{k}^{(u,v)} + 0.5(1 + \sum_{\forall i \in [m]} |{A_i}^{(u,v)}|))Q^{(u,v)} + \sum_{\forall i \in [m]} {A_i}^{(u,v)}\varepsilon_i^{(u,v)} &. \end{aligned}$$

2. If ${Y_j}^{(u,v)} < k$, then

$$\tilde{Y}_j^{(u,v)} < \tilde{k}^{(u,v)} Q^{(u,v)} + 0.5 Q^{(u,v)} (1 + \sum_{\forall i \in [m]} |{A_i}^{(u,v)}|) + \sum_{\forall i \in [m]} {A_i}^{(u,v)} \varepsilon_i^{(u,v)} \ .$$

3. If ${Y_j}^{(u,v)} > k$, then

$$\tilde{Y}_j^{(u,v)} > \tilde{k}^{(u,v)} Q^{(u,v)} - 0.5 Q^{(u,v)} (1 + \sum_{\forall i \in [m]} |{A_i}^{(u,v)}|) + \sum_{\forall i \in [m]} {A_i}^{(u,v)} \varepsilon_i^{(u,v)} \ .$$

Let $\tau^{(u,v)}$ be a non-negative real number such that $-\tau^{(u,v)} \leq \sum_{\forall i \in [m]} {A_i}^{(u,v)} \varepsilon_{j,i}^{(u,v)} \leq \tau^{(u,v)}$ for all ${G_j}^{(u,v)}$, and assume $\varepsilon_i^{(u,v)}$ has a normal distribution with zero-mean. Then for large m, $\sum_{\forall i \in [m]} {A_i}^{(u,v)} \varepsilon_i^{(u,v)}$ is expected to have

$$\sum_{\forall i \in [m]} {A_i}^{(u,v)} \varepsilon_i^{(u,v)} \approx 0 \ . \tag{7.13}$$

That is, the calculation errors tend to cancel each other. This is an advantage of using larger sums. However, as will be seen in Section 7.4.2, using larger sums has other problems.

Verification

The verification process uses an error tolerance interval and as long as $\tilde{Y}_j^{(u,v)}$, calculated from the image, is within this interval around $U(N)$, calculated from the recovered feature codes, the verification is successful. For a given quality level ℓ, an *error tolerance level* $E^{(u,v)}$ can be calculated (see below equation (7.14)), where $E^{(u,v)} \in \mathbb{R}$ and $E^{(u,v)} \geq 0$.

Verification proceeds as follows.

	Algorithm 3 : Verification of $\tilde{Y}_j^{(u,v)}$
1:	Set a tolerance level to $E^{(u,v)}$.
2:	Calculate $\tilde{Y}_j^{(u,v)} = \sum_{\forall i \in [m]} {A_i}^{(u,v)} \tilde{F}_{i,j}^{(u,v)}$.
3:	Obtain $U(N)$ from the feature code (equations (7.10) and (7.11)).
4:	if $U(N) - E^{(u,v)} < \tilde{Y}_j^{(u,v)} < U(N) + E^{(u,v)} + ({F_{MAX}}^{(u,v)} - {F_{MIN}}^{(u,v)}) 2^{-N}$ (From equation (7.12), the difference $\lvert {Y_j}^{(u,v)} - U(N) \rvert$ can be as large as $({F_{MAX}}^{(u,v)} - {F_{MIN}}^{(u,v)}) 2^{-N}$.)
5:	verification of $\tilde{Y}_j^{(u,v)}$ is successful.
6:	else
7:	the image is tampered.

Algorithm 3 is repeated for all feature codes and if it succeeds for all feature codes then the image is considered authentic. To choose $E^{(u,v)}$ for tolerance to quality level ℓ and assuming the computation error $\tau^{(u,v)}$ as defined in Section 7.4.1 and Theorem 4, we have

$$E^{(u,v)} \leq 0.5 {Q^{(u,v)}}_{\ell} \sum_{\forall i \in [m]} |{A_i}^{(u,v)}| + \tau^{(u,v)} \ .$$

${Q^{(u,v)}}_{\ell}$ in the above is the maximum value of $Q^{(u,v)}$ corresponding to the lowest JPEG quality level that must be accepted by the system. In practice, because of the reasons described in Section 7.4.1, smaller values of $E^{(u,v)}$ can be chosen.

Theorem 5 *(soundness) The authentication system described above tolerates image degradation due to JPEG compression to a designated quality level ℓ.*

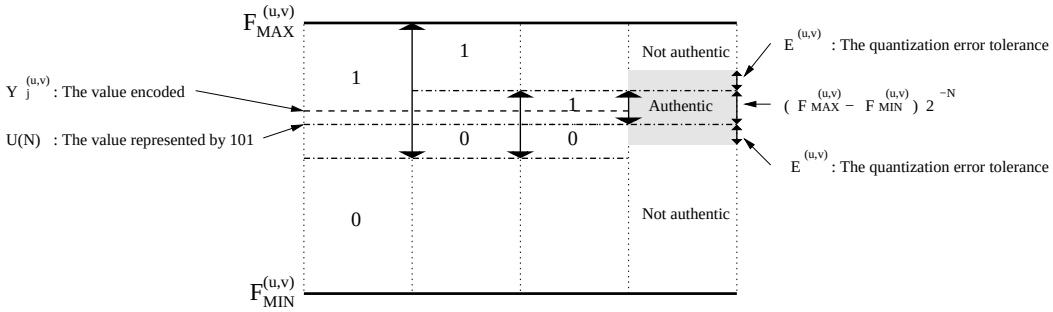


Figure 7.9: Encoding of ${Y_j}^{(u,v)}$ and error tolerance.

SARI System

The SARI system is a special case of the proposed system where $m = 2$ and ${A_1}^{(u,v)}$, ${A_2}^{(u,v)}$ are 1 and -1, respectively, and so $\sum_{\forall i \in [m]} |{A_i}^{(u,v)}| = 2$.

From Theorem 3, the following is true.

1. If ${Y_j}^{(u,v)} = k$, then $\tilde{Y}_j^{(u,v)}$ can take one of three possible values,

$$\tilde{F}_1^{(u,v)} - \tilde{F}_2^{(u,v)} = (\tilde{k}^{(u,v)} - 1)Q^{(u,v)} \ , \tilde{k}^{(u,v)}Q^{(u,v)} \ , \text{or} (\tilde{k}^{(u,v)} + 1)Q^{(u,v)}$$

2. If ${Y_j}^{(u,v)} < k$, then

$$\tilde{F}_1^{(u,v)} - \tilde{F}_2^{(u,v)} < (\tilde{k}^{(u,v)} + 1)Q^{(u,v)}$$

3. If ${Y_j}^{(u,v)} > k$, then

$$\tilde{F}_1^{(u,v)} - \tilde{F}_2^{(u,v)} > (\tilde{k}^{(u,v)} - 1)Q^{(u,v)}$$

Theorem 6 *Assuming a fixed threshold* $k \in \mathbb{R}, \forall u, v$, *define* $\tilde{k}^{(u,v)} \equiv rint(\frac{k}{Q^{(u,v)}})$.

if ${\Delta F_{p,q}}^{(u,v)} > k$,

$$\Delta \tilde{F}_{p,q}^{(u,v)} \geq \begin{cases} \tilde{k}^{(u,v)} Q^{(u,v)}, & \frac{k}{Q^{(u,v)}} \in \mathbb{Z} \\ (\tilde{k}^{(u,v)} - 1) Q^{(u,v)}, & \notin \mathbb{Z} \end{cases} \tag{7.14}$$

else if ${\Delta F_{p,q}}^{(u,v)} < k$,

$$\Delta \tilde{F}_{p,q}^{(u,v)} \leq \begin{cases} \tilde{k}^{(u,v)} Q^{(u,v)}, & \frac{k}{Q^{(u,v)}} \in \mathbb{Z} \\ (\tilde{k}^{(u,v)} + 1) Q^{(u,v)}, & \notin \mathbb{Z} \end{cases} \tag{7.15}$$

else if ${\Delta F_{p,q}}^{(u,v)} = k$,

$$\Delta \tilde{F}_{p,q}^{(u,v)} = \begin{cases} \tilde{k}^{(u,v)} Q^{(u,v)}, & \frac{k}{Q^{(u,v)}} \in \mathbb{Z} \\ \tilde{k}^{(u,v)} Q^{(u,v)} \ or \ (\tilde{k}^{(u,v)} \pm 1) Q^{(u,v)}, & \notin \mathbb{Z} \end{cases} . \tag{7.16}$$

From Theorem 3, the following is true.

1. If ${Y_j}^{(u,v)} = k$, then

$$(\tilde{k}^{(u,v)} - 1.5) Q^{(u,v)} < \tilde{F}_1^{(u,v)} - \tilde{F}_2^{(u,v)} < (\tilde{k}^{(u,v)} + 1.5) Q^{(u,v)}$$

but $\tilde{F}_1^{(u,v)} - \tilde{F}_2^{(u,v)}$ is an integer and so

$$\tilde{F}_1^{(u,v)} - \tilde{F}_2^{(u,v)} = (\tilde{k}^{(u,v)} - 1) Q^{(u,v)},$$

$$\tilde{F}_1^{(u,v)} - \tilde{F}_2^{(u,v)} = \tilde{k}^{(u,v)} Q^{(u,v)}, \text{or}$$

$$\tilde{F}_1^{(u,v)} - \tilde{F}_2^{(u,v)} = (\tilde{k}^{(u,v)} + 1) Q^{(u,v)}$$

2. If ${Y_j}^{(u,v)} < k$, then

$$\tilde{F}_1^{(u,v)} - \tilde{F}_2^{(u,v)} < (\tilde{k}^{(u,v)} + 1.5) Q^{(u,v)}$$

but $\tilde{F}_1^{(u,v)} - \tilde{F}_2^{(u,v)}$ is an integer and so

$$\tilde{F}_1^{(u,v)} - \tilde{F}_2^{(u,v)} < (\tilde{k}^{(u,v)} + 1) Q^{(u,v)}$$

3. If ${Y_j}^{(u,v)} > k$, then

$$\tilde{F}_1^{(u,v)} - \tilde{F}_2^{(u,v)} > (\tilde{k}^{(u,v)} - 1.5) Q^{(u,v)}$$

but $\tilde{F}_1^{(u,v)} - \tilde{F}_2^{(u,v)}$ is an integer and so

$$\tilde{F}_1^{(u,v)} - \tilde{F}_2^{(u,v)} > (\tilde{k}^{(u,v)} - 1) Q^{(u,v)}$$

These relationships are the same as those in [58]. If the same $k^{(u,v)}$ is used, the system will produce the same values for feature codes as those generated by the system described in Section 7.4.1. For example, $Y_j^{(u,v)} = 11$, $n = 4$ and the sequence of $k^{(u,v)}$ is $k^{(u,v)} = 16, 8, 12, 10$ then the bits generated are `0101` which is the same as the linear combination scheme.

7.4.2 Designing a Message Authentication Code

The main requirement for a cryptographic hash function is collision intractability without using a secret key. In our proposed system, if all design parameters of the systems are public, it will be easy to construct two images that have the same feature codes and so finding a collision will be easy. To provide collision security, some of the parameters of the system must be used as secret key. Hence the system is effectively *a keyed hash function* or a *message authentication code*.

The system's parameters are the following.

1. The number of blocks in a group, m, and the composition of the groups, ${G_j}^{(u,v)}$, $j \in \{1, 2, 3, ..., g\}$.
2. Coefficients of the linear combination ${A_i}^{(u,v)}$, $i \in \{1, 2, 3, ..., m\}$.
3. The set of protected frequencies (u, v).
4. The precision (number of bits) $N^{(u,v)}$ allocated to feature codes.
5. The error tolerance $E^{(u,v)}$.

To construct a message authentication code some of the above parameters must be kept as shared secrets between the sender and the receiver. The aim is to minimize the length of secret key while maintaining a high security and ensure that the key bits are all affecting the value of the MAC.

After careful examination of possible subsets of parameters (details omitted), we propose the following information to be the key information.

- The group composition ${G_j}^{(u,v)}$, $j \in \{1, 2, 3, ..., g\}$
- The linear combination coefficients ${A_i}^{(u,v)}$, $i \in \{1, 2, 3, ..., m\}$.

Other parameters such as the group size, precision of the feature codes, error tolerance and the choice of protected frequencies will be public. We assume that *the secret key*

is changed with every image. That is, all system parameters including the composition of groups are public but the correspondence between image blocks and their abstract representation will be the secret key. This means that the size of the key space for the group composition is $\wp!$.

For precision of the feature codes and error tolerance, the subsections *Feature Code*, *Finding the Tolerance Interval*, *Quantization Error* and *Computation Error* in Section 7.4.1 give more details on how to choose the key parameters.

Selecting the Linear Combination Coefficients

Let ${A_{MIN}}^{(u,v)}$ and ${A_{MAX}}^{(u,v)}$ be two integers denoting the maximum and the minimum value of a linear combination coefficient, and assume ${A_i}^{(u,v)}$ is randomly chosen from the interval $[{A_{MIN}}^{(u,v)}, {A_{MAX}}^{(u,v)}]$. Hence, there are ${A_{MAX}}^{(u,v)} - {A_{MIN}}^{(u,v)} + 1$ possible values for ${A_i}^{(u,v)}$.

The value of ${A_i}^{(u,v)}$ determines the protection level of the coefficient ${F_{i,j}}^{(u,v)}$ and can be measured in terms of *the number of protected bits.*

In general, a DCT coefficient modified by α is multiplied by ${A_i}^{(u,v)}$ and the larger ${A_i}^{(u,v)}$ means that the modification is scaled up by a larger factor. This means that the same amount of change is scaled by $\frac{{A_a}^{(u,v)}}{{A_b}^{(u,v)}}$ for $\tilde{F}_{a,j}^{(u,v)}$ compared to $\tilde{F}_{b,j}^{(u,v)}$. In other words $\tilde{F}_{a,j}^{(u,v)}$ has $\log_2 \frac{{A_a}^{(u,v)}}{{A_b}^{(u,v)}}$ more protected bits compared to $\tilde{F}_{b,j}^{(u,v)}$ if ${A_a}^{(u,v)} > {A_b}^{(u,v)}$. For example, if ${A_a}^{(u,v)} = 2{A_b}^{(u,v)}$, the bit string representing $\tilde{F}_{a,j}^{(u,v)}$ has 1 more protected bit than $\tilde{F}_{b,j}^{(u,v)}$ in the sum $\sum_{i=1}^{m} {A_i}^{(u,v)} \tilde{F}_{i,j}^{(u,v)}$.

Hence if all regions of the image have the same significance, then ${A_i}^{(u,v)}$ must be chosen close to each other. On the other hand if parts of the image need higher protection, their corresponding coefficients must be chosen to have higher values.

Determining the Coefficient Range

To determine the range of the linear combination coefficients, we must estimate the effect of changing DCT coefficients in pixel domain. Our experimental results show that modifying a DCT coefficient by ± 7 results in the value of pixels changing by ± 1, and modifying it by ± 15 results in the values of pixels changing by ± 3. Since a ± 1 change in the pixel domain is visually insignificant, the relative value of these coefficients are chosen as $\frac{{A_a}^{(u,v)}}{{A_b}^{(u,v)}} \leq 2^3$. This results in the effect of scaling to be within ± 1 in the pixel domain and so

$$ {A_{MIN}}^{(u,v)} \leq {A_i}^{(u,v)} \leq 8{A_{MIN}}^{(u,v)} \ . $$

To choose ${A_{MIN}}^{(u,v)}$, we note that the range of ${A_i}^{(u,v)}$ is given as

$$\begin{aligned} \rho &= 8{A_{MIN}}^{(u,v)} - {A_{MIN}}^{(u,v)} + 1 \\ &= 7{A_{MIN}}^{(u,v)} + 1 \end{aligned}$$

and for larger ${A_{MIN}}^{(u,v)}$, the number of possible values of ${A_i}^{(u,v)}$ increases, and so the size of the key space increases. Now if an attacker wants to modify a block, he has to find other blocks in the same group and their corresponding ${A_i}^{(u,v)}$ to make the compensating modification. In the case of using an exhaustive search to find ${A_i}^{(u,v)}$, the size of key space and $\wp$ determine the cost of attack and so larger ${A_{MIN}}^{(u,v)}$ will require more computation. However, larger ${A_{MIN}}^{(u,v)}$ means longer feature codes will be generated.

The number of bits for ${A_i}^{(u,v)}{F_{i,j}}^{(u,v)}$, that is $|{Y_j}^{(u,v)}|$, is given by

$$|{Y_j}^{(u,v)}| = \log_2(m{A_{MAX}}^{(u,v)}{F_{MAX}}^{(u,v)}) \ . \tag{7.17}$$

Construction of Groups

There are $\wp$ blocks in the image and the aim is to construct groups of size m that together cover the whole image. The group composition can be described by an incidence matrix, where rows correspond to groups and columns correspond to blocks. The matrix entries are 0 and 1 with 1 in the (i, j) position showing that group j includes block i. Given a matrix as above, a random labeling of image blocks can be used to map groups into the image block. The number of groups and their composition determine the length and security of the MAC and so must be chosen carefully.

Number of Groups

To determine the number of groups, the following should be considered. (We assume groups have the same size, although it is possible to have varying sizes when some parts of the image need higher protection.) A larger size for a group gives more flexibility to an attacker to make his modification of a chosen block imperceptible as he can spread the compensating change over a large number of blocks.

In general larger groups result in less sensitivity to change in a block because the distribution of DCT coefficients is known to be a generalized Gaussian [25, 47, 54], the density function of which is given by [26]

$$p(x) = \left[\frac{\nu\eta(\nu,\sigma)}{2\Gamma(1/\nu)}\right] exp(-[\eta(\nu,\sigma)|x|]^{\nu})$$

where

$$\eta(\nu,\sigma) = \sigma^{-1}\left[\frac{\Gamma(3/\nu)}{\Gamma(1/\nu)}\right]^{1/2}$$

and summing a large number of coefficients will even out the local variations in the image. We noted that larger group size reduces the average calculation time and quantization error, and allows the choice of a narrower error tolerance interval compared to the sum of these errors. Larger means less groups are required to cover the whole image and so a shorter MAC will be produced. The number of blocks in a group must be chosen by taking these conflicting requirements into account. Typical values are 8, 16, and 32.

Choosing Groups

If the groups are disjoint, the change to a block will stay local and it will be easier to find the group members using an attack similar to Algorithm 1 in Section 7.3.1 although the attacker has to find more than one block in the same group and so the cost is larger. Here the attacker needs to modify blocks in one group such that the corresponding feature code is unaffected. By linking the groups together, that is, by allowing blocks to belong to more than one group, the attacker will have a more difficult task as the change in one block will affect many more blocks in the image. For example by requiring each block to belong to two groups, a change in a block will affect two groups (two feature codes) and to compensate this change at least one more block in each group needs to be modified. Assuming the two groups intersect with one block, one of the two blocks is not in the intersection of the two groups and so the effect spreads to other feature codes.

Summarizing the above discussion, we give two conditions that need to be satisfied by groups.

Condition 1 Each block belongs to λ groups where $\lambda \geq 2$.

Condition 2 Two groups intersect in 1 block.

These conditions can be optimally met by a combinatorial structure called *Steiner system.*

A Steiner system is defined as follows.

Definition 1 *A Steiner system is a set X of v points, and a collection of subsets of X of size k, such that any t points of X are in exactly one of the subsets [19].*

Using a Steiner system ensures that all groups have the size (v), each block appears in exactly r groups and two groups have $t-1$ blocks in common.

It is known that Steiner systems $S(t,k,v)$ exist for $S(2,n+1,n^2+n+1)$ where n is the order of a projective plane and $S(2,q,q^d)$ where q is a prime power [119].

Relaxing the conditions that need to be met by groups allows us to use a wider range of combinatorial constructions. In Section 7.4.3 we give an algorithm that can generate groups satisfying **Condition 1** and **Condition 2** for any image size.

7.4.3 Constructing Groups

The above two methods restrict the choice of m and g. In the following, we describe a method that has less restrictions.

Let each block belong to two groups, m be an even number and $m < g-1$ (a restriction). Let M be an $\frac{m}{2} \times g$ matrix and let $x_{j,i}$ be the element in row j and column i of M, where $i \in \{1,2,...,g\}$, $j \in \{1,2,...,\frac{m}{2}\}$. We construct M as,

$$\begin{pmatrix} 1 & 1 & 0 & 0 & \dots & 0 \\ 1 & 0 & 1 & 0 & \dots & 0 \\ 1 & 0 & 0 & 1 & \dots & 0 \\ \vdots & & & & & \vdots \end{pmatrix}$$

In other words, $x_{j,i}$ is given as,

$$x_{j,i} = \begin{cases} 1 & \text{if } i = 0 \text{ or } i = j+1 \\ 0 & \text{if } i \neq 0 \text{ and } i \neq j+1 \ . \end{cases}$$

Let the right *rotation* of a row be defined as moving $x_{j,i}$ to position $(i+1 \mod g)$. There are g possible rotations of a row given by, $(x_{j,(1+k) \mod g}, x_{j,(2+k) \mod g}, \dots, x_{j,(g+k) \mod g})$, $k \in \{0,1,...,g-1\}$. For example, rotations of the 1st row in M are given by, $(1,1,0,0,\dots,0)$, $(0,1,1,0,\dots,0)$, ..., and $(1,0,0,0,\dots,1)$. Then for all j,k, we have $\sum_{i=1}^{g} x_{j,(i+k) \mod g} = 2$ and for all j,i, we have $\sum_{k=0}^{g-1} x_{j,(i+k) \mod g} = 2$. We randomly assign a block to each row and include the block in G_i if $x_{j,i} = 1$.

If we repeat the above procedure for $\frac{m}{2}$ rows in M, we will obtain $g \times \frac{m}{2} = \wp$ rows. Each row will be randomly assigned to a block and so each group G_i will have $2 \times \frac{m}{2} = m$ blocks. This method guarantees that all blocks are linked.

We choose $\frac{m}{2}$ to be the number of rows in M to prevent the rotation of different rows from the colliding result. If two rows where ith and $i+a$th columns are 1 and the ith and $i+b$th columns are 1 are rotated, they will collide if $a+b=g$. Since this can

be avoided if $a, b < \frac{g}{2}$, we choose the number of rows in M as $\frac{m}{2}$ and set the restriction to be $m < g - 1$, as given above.

The procedure is as follows.

	Algorithm 4 : Construction of groups
1:	$G_i = \phi$, for all $i \in \{1, 2, ..., g\}$.
2:	A set P includes all blocks.
3:	For jth row in M where $j \in \{1, 2, ..., \frac{m}{2}\}$
4:	Generate g rotated rows from jth row in M.
5:	For each rotated row
6:	Randomly choose a block in P.
7:	If ith column of the rotated row is 1
8:	Include the block in G_i.
9:	Remove the block from P.

7.4.4 Evaluation of the MAC

We evaluate the security and efficiency of the proposed MAC system. For security, we propose an attack model that corresponds to a likely real life application of the system, and show that it is infeasible to construct a forged image, MAC pair in that model.

For efficiency we consider the length of the MAC and the time spent on generating a MAC.

Security

In this section we propose a model for evaluating the security of a MAC system for images. All previously proposed systems use an ad-hoc approach with no clear definition of attacks, and capabilities and goals of the attacker.

We consider the following application scenario.

An attacker owns a client decoder and aims at constructing a fraudulent image after (or before) receiving an authentic one.

The attacker succeeds if he can construct a forged image, MAC pair that (i) passes the verification test, and (ii) does not have any visual artifacts that make it suspicious.

We assume the attacker does not have access to the decoder key (black box) but can query the decoder with other image, MAC pairs. The attacker can supply image, MAC pairs to the decoder and receive a true response if the pair is valid and false otherwise.

We assume that *the authentication key changes with each original image.* The same MAC is used for all images generated from the original image by compressing the original image with different quality level (or re-compressing the compressed image). This is a reasonable assumption because as noted before regions of interest in images are different and so the multipliers should be adapted to the protection that is required for the particular image. With this assumption, although the attacker can access the MAC generation oracle many times but since the key changes with each image, then the attacker cannot gain any new information about the key by using multiple queries to the MAC generation oracle. Hence we only consider the information that the attacker will gain by interacting with the verification oracle.

This attack scenario corresponds to the case that a malicious user having a decoder tries to impersonate a server that sells authenticated images. This model and assumption match most of the existing terminal architectures that support digital rights management and assume trusted hardware for decoding of data [15, 48].

We assume that attacker knows the system parameters *i*) the range of $A_i^{(u,v)}$, *ii*) the coefficients (u, v) that are protected, *iii*) the number of groups g and hence the number of blocks m in a group, and *iv*) the feature code.

The attacker does not know *i*) the blocks in each group, and *ii*) the coefficients $A_i^{(u,v)}$, and his aim is to either construct a valid image-MAC pair, or modify an image so that its MAC value does not change. These two attacks correspond to impersonation and substitution attacks in authentication codes with the difference that the attacker has a decoder box but does not know the key. It is straightforward to see that the success chance of an attacker in a substitution attack, that is modifying an authenticated image, is higher than trying to construct a pair of a valid image and MAC and so we only consider the substitution attack. In the following we consider the computation cost of possible attacks.

Cost of Modifying a Block

To modify a chosen block the attacker has to find other blocks in the same group and modify them such that the change to the chosen block is compensated. He also has to find the linear combination coefficients corresponding to those blocks.

To simplify the analysis first we assume that each coefficient belongs to only one group. The attacker does the following.

1. Add δ to DCT coefficient in block p_{l_1} which he intends to modify.

2. Repeat until the verification succeeds.

(a) Choose a block p_{l_2}, $l_2 \in \{1, 2, ..., g\}$, $l_1 \neq l_2$.

(b) Subtract $\frac{{A_{i_2}}^{(u,v)}}{{A_{i_1}}^{(u,v)}}\delta$ from the DCT coefficient in block l_2 to cancel out the modification of block l_1.

(c) Input the modified image and the authenticator to the verification oracle. The verification will succeed if the choice of the blocks $p_{l_1}, p_{l_2} \in {G_j}^{(u,v)}$ and coefficients ${A_{i_1}}^{(u,v)}$ and ${A_{i_2}}^{(u,v)}$ are correct.

The attacker does not know the mapping between groups and blocks and the coefficients ${A_{i_1}}^{(u,v)}$ and ${A_{i_2}}^{(u,v)}$ and so he has to try all possible $l_2 \in \{1, 2, ..., g\}$, $l_1 \neq l_2$ and all possible values of ${A_{i_1}}^{(u,v)}$ and ${A_{i_2}}^{(u,v)}$. Let ${A_i}^{(u,v)}$ be independently chosen in the range $[{A_{MIN}}^{(u,v)}, {A_{MAX}}^{(u,v)}]$. Then the number of possible combinations of ${A_{i_1}}^{(u,v)}$ and ${A_{i_2}}^{(u,v)}$ is $({A_{MAX}}^{(u,v)} - {A_{MIN}}^{(u,v)} + 1)^2$. The attacker tries $\wp - 1$ blocks for the above combination and so the number of the trials C is,

$$C = (\wp - 1)({A_{MAX}}^{(u,v)} - {A_{MIN}}^{(u,v)} + 1)^2 \ . \tag{7.18}$$

For example, if $\wp = 4096$ and ${A_{MAX}}^{(u,v)} - {A_{MIN}}^{(u,v)} = 8$, $C = 2^{12}(2^3)^2 \approx 2^{18}$.

Assume a block is in two groups. Now the number of blocks that the attacker has to modify increases because

- The coefficient ${A_{i_1}}^{(u,v)}$ of a chosen block can take ${A_{MAX}}^{(u,v)} - {A_{MIN}}^{(u,v)} + 1$ values and so using exhaustive search for each ${A_{i_1}}^{(u,v)}$, requires ${A_{MAX}}^{(u,v)} - {A_{MIN}}^{(u,v)} + 1$ experiments. Each DCT coefficient belongs to two groups. The position i of the DCT coefficient in $\sum_{\forall i \in [m]} {A_i}^{(u,v)} {F_{i,j}}^{(u,v)}$ in each group is independently chosen and so will be different in the two groups and so the corresponding linear combination coefficients in the two groups will be different. Hence, to find two linear combination coefficients for one DCT coefficient, $({A_{MAX}}^{(u,v)} - {A_{MIN}}^{(u,v)} + 1)^2$ experiments are required.

- Two DCT coefficients, each belonging to one of the two groups, have to be modified to compensate for the above modification of the chosen block and so their corresponding four linear combination coefficients need to be found. The cost will be $({A_{MAX}}^{(u,v)} - {A_{MIN}}^{(u,v)} + 1)^3$.

- If all groups are linked, the attacker needs to modify all blocks and so he has to find the linear combination coefficients for all blocks. The cost is $({A_{MAX}}^{(u,v)} - {A_{MIN}}^{(u,v)} + 1)^{gm}$. Also he has to find the mapping between groups and blocks to add or subtract a value for the modification. The values and the required

operations (addition or subtraction) depend on which group a block belongs to. The cost of finding blocks in G_j is $C(\wp-(m-1)(j-1),m)$ However, the attacker needs to find blocks of G_j for all j simultaneously and so the cost is $\sum_{j=1}^{g} C(\wp-(m-1)(j-1),m)$.

For example, if $\wp = 4096$ and ${A_{MAX}}^{(u,v)} - {A_{MIN}}^{(u,v)} = 8$, then $gm = 2\wp$ because each block belongs to two groups and so the cost of finding the linear combination coefficients is $C = 2^{10}(2^8)^2 \approx 2^{26}$. The attacker also needs to find blocks of all groups.

As described above, the compensating modification propagates to a large number of blocks and so becomes more likely to be detected as more and more blocks with unknown multipliers must be modified.

Length of the Signature

The length of the MAC is given by $L = g\sum_{(u,v)\in\Pi} N^{(u,v)}$ where Π is the set of protected frequencies. We note that,

1. To protect all blocks the union of blocks in all groups must cover the whole image. That is $\cup_i G_i = P$.

2. The length of the MAC is proportional to the number of groups. This suggests having fewer groups but fewer groups means larger number of blocks in a group. (See Section 7.4.2 **Construction of groups**, **Number of groups**, and **Choosing groups** for further discussion.)

3. The protected frequencies can be image dependent. An image with not much detail does not need to have the high frequency components protected.

4. The number of bits allocated to the feature codes corresponding to ${Y_j}^{(u,v)}$ is determined by the compression level that must be tolerated. If only high quality images must be acceptable, then more bits must be allocated to the feature codes.

Decreasing the MAC size will increase the probability of false acceptance. Example MAC sizes for 512×512 gray scale image with all frequencies protected are 24K bytes ($m = 8$) and 16K bytes ($m = 16$).

Implementation

The proposed system can be integrated into the JPEG compression and decompression system as shown in Figure 7.7 and 7.8.

The MAC generation system inputs 8×8 DCT blocks and the secret key, together with parameters such as g and (u, v) and outputs the MAC. The MAC verification system inputs 8×8 blocks of dequantized DCT coefficients, the MAC and the secret

key, together with the MAC generation parameters and outputs a true or false result. The decompressed image can be also used for the verification. In this case, the image needs to be transformed using an 8×8 DCT instead of the dequantized DCT coefficients from the JPEG decompression system.

The important properties of the system are :

- Computing feature codes are independent from each other and so can be made in parallel. This means that the effective computation time for the MAC is equal to computing a single feature code and so is very fast.
- The scheme uses an 8×8 DCT. That is, hardware and software implementation of system can be made by only a small change to the JPEG implementation.

We implemented the systems and performed a number of experiments to verify our theoretical results. More details on the experiments are given in Section 7.4.5. Our experiments show that local and global modifications can be effectively detected for MAC sizes of 24K and 16K bytes. However small modifications may not be detected with the smaller size MAC.

7.4.5 Experiments

In our experiments the groups were chosen as follows.

- The blocks in each group were randomly chosen so that each block was included in at least two groups.
- Each group consisted of the same number of blocks.
- $A_i^{(u,v)} = 1$ for all i. This means all coefficients of the same frequency are equally protected. This will be used if the protection level is chosen independent of image contents, i.e. the same protection level is used for various images.

The image was the 512×512 gray scale `lena`. Two different types of modification were made on the image,

i) local modification on a small region and *ii*) global modification of the whole image.

For *i*), the modification made was to add a beauty mark to `lena` by adding a 3×3 pixel dot below Lena's left eye as shown in Figure 7.10. For *ii*), the original image was modified using a median filter with 3×3, 5×5, 7×7 and 9×9 window sizes, as shown in Figure 7.11.

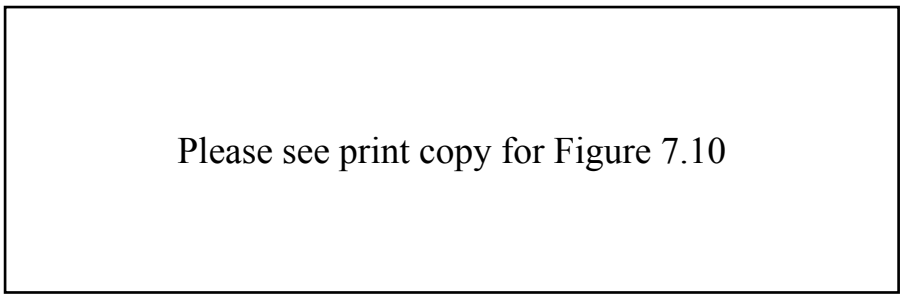
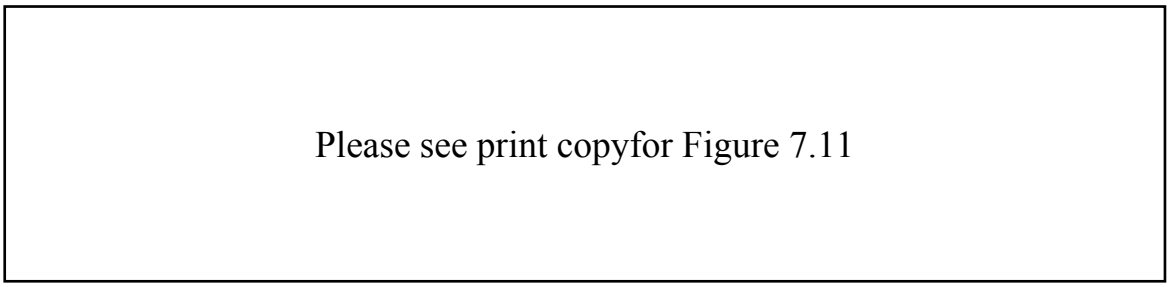


Figure 7.10: Lena with a beauty mark (left) and close-up of the modified region (right).



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

Figure 7.11: Lena using a median filter. 3×3 (a), 5×5 (b), 7×7 (c) and 9×9 (d) window sizes.

The original image was used for the feature code generation. The MAC sizes for $m = 8, 16, 32, 64$ are shown in Table 7.1. For the `lena` image with a beauty mark, quality level 95%, 75% and 50% JPEG compressed images were generated from the modified image. For the experiments of the median filtered images, the filtered images were created from the original image.

The numbers of groups were 1024, 512, 256 and 128 and the number of blocks in a group m were 8, 16, 32 and 64, respectively. The feature code included all the frequencies from DC to AC_{63}. The precisions (i.e. the number of bits) corresponding to the 8×8 DCT feature codes in positions (u, v) for $m = 8$ are shown in Table 7.2. For $m = 16, 32, 64$, the values in the table were increased by 1,2 and 3, respectively. This is because if m is doubled, it requires one more bit to represent the sum of $2m$ coefficients, compared to the case with m coefficients.

They are chosen by analyzing the compressed images (quality=95, 75, 50 %) so that all these images will be authenticated, that is, the largest quantization error level for those images was chosen. They correspond to the 8×8 DCT positions (u, v).

The tolerance values are the same scale as the (sum of) dequantized coefficient values. In the same scale, the ranges of a DC coefficient and an AC coefficient are

$[0, 2048)$ and $[-1024, 1024)$, respectively (i.e. 256 (pixel value) $\times$8(scaling factor)). The DC quantization error interval for the above case is $[-0.5 \times 16 \times 16, 0.5 \times 16 \times 16]$, where $Q^{(u,v)} = 16$ for 50% quality level, $m = 16$ and ${A_{i,j}}^{(u,v)} = 1$, and so it is $[-128, 128]$. Compared with the value of 128, the quantization error tolerance of 60 is about half.

The results of the verification of *i*) and *ii*) are shown in Table 7.4. The upper table in Table 7.4 shows the number of DCT coefficients in a group, the compression quality used, the verification result and the number of groups which failed the verification. For $m = 8, 16$, the modification was detected. The reason of this would be that the larger m means that the larger sum of quantization errors and so the errors are too large compared with the change by the modification, which is not very large, as shown in Table 7.3. This suggests that protecting an image against a small change, m cannot be large. The lower table in Table 7.4 shows the number of DCT coefficients in a group, the median filter window size, the verification result and the number of groups which failed the verification, and all the filtered images failed the verification. The reason of this is that the modification by the filter spreads over the whole image and the amount of changes is large. This can be seen from the number of blocks which failed the verification in the table. Compared with the beauty mark modification case, it is significantly larger.

Table 7.1: Number of coefficients per group and the MAC size.

# of coef / group	MAC size
8	24576 bytes
16	16384 bytes
32	10240 bytes
64	6144 bytes

Table 7.2: Precisions for linear sums ($m = 8$).

5	5	5	4	4	4	3	3
5	5	4	4	4	3	3	3
5	4	4	4	3	3	3	2
4	4	4	3	3	3	2	2
4	4	3	3	3	2	2	2
4	3	3	3	2	2	2	1
3	3	3	2	2	2	1	1
3	3	2	2	2	1	1	1

Table 7.3: DCT coefficients of modified 8×8 block of `lena` (top) and those of the original (bottom).

865.9	13.0	11.6	5.5	-16.4	-5.5	5.4	-2.6
-80.2	-19.2	6.6	-9.3	-20.7	-15.0	1.6	4.0
-22.1	-1.5	1.9	-1.7	-11.8	-10.2	1.0	1.6
-1.2	-1.6	6.1	2.7	-2.9	0.2	-4.5	-6.2
5.9	0.4	-1.1	-0.9	0.6	4.3	-2.9	-4.8
-3.6	3.4	5.7	-3.4	0.0	2.4	-0.9	2.5
3.1	-1.6	0.2	-1.7	1.2	2.3	-3.9	-3.3
-3.9	0.9	-1.8	3.5	1.9	-1.5	-3.0	-2.3

805.6	57.6	38.7	-44.2	3.1	-3.0	15.2	-22.3
-35.5	-52.6	-12.9	27.4	-35.8	-16.0	-6.1	18.5
4.4	-20.6	-11.2	20.5	-19.0	-12.9	-3.1	10.9
-50.2	34.8	27.9	-37.2	12.6	1.8	5.0	-23.5
25.4	-15.0	-7.7	14.0	-7.1	6.5	-8.6	2.5
-1.9	2.9	3.5	-1.8	1.4	0.3	-1.1	4.2
14.3	-9.6	-5.9	9.2	-3.1	0.1	-1.8	-3.1
-24.5	15.4	9.5	-15.4	8.2	2.5	-4.0	-5.9

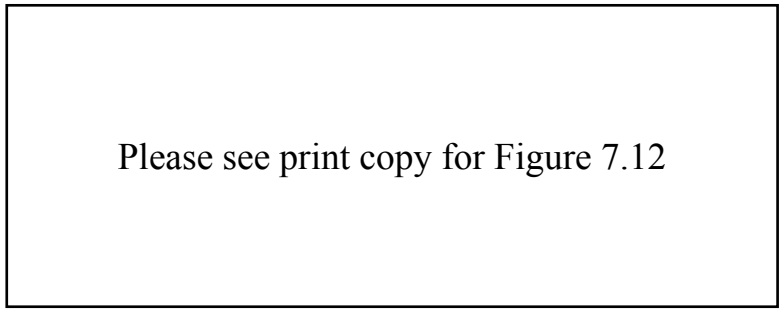


Figure 7.12: Close up of the right eye of lena. The center 8×8 block is at position (264,272) modified by a median filter with 9×9 window sizes.

7.4.6 Quantization Error Distribution

The following experimental results show the JPEG quantization errors with various quality levels.

The method of obtaining quantization errors for a JPEG compressed image is as follows.

1. Using the `cjpeg` command, create a JPEG compressed image from the original `pgm` image.

2. Decode the compressed image using the `djpeg` command and obtain the decompressed image in `pgm` format.

3. Then DCT transform both the original and the de-compressed images. The error is obtained by finding the difference between two corresponding DCT coefficients in the two images.

The distribution of the errors was obtained by integer-rounding the error values and then counting the number of each error value.

The quality levels used for the image `lena` were 95%, 75%, 50% and 25%.

The graphs in Figure 7.13, 7.14, and 7.15 show the results. Each graph shows the distribution of a particular coefficients over the whole image when the image is compressed and decompressed to the given quality values. The x axis shows the quantization errors, the y axis is the quality level and the z axis is the frequencies of the error values. The positions of the graphs correspond to the frequencies in 8×8 DCT coefficient matrix, i.e. the top left corner corresponds to the DC coefficients, and the bottom right corner is the distribution of AC63. The graphs show Gaussian like distribution with zero mean and the lower frequency has larger variance. This means that a sum of errors will also have Gaussian like distribution with zero mean but with a smaller variance (See Section 7.4.1).

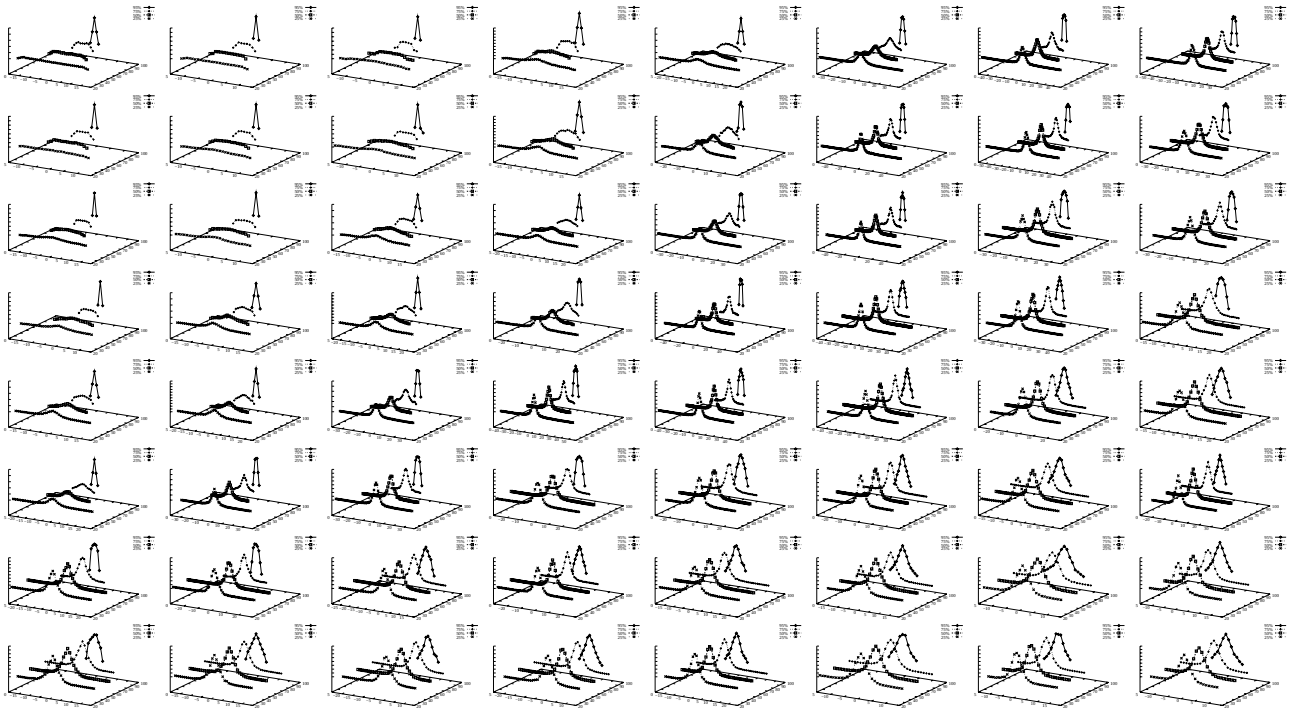


Figure 7.13: Distribution of errors : `lena`.

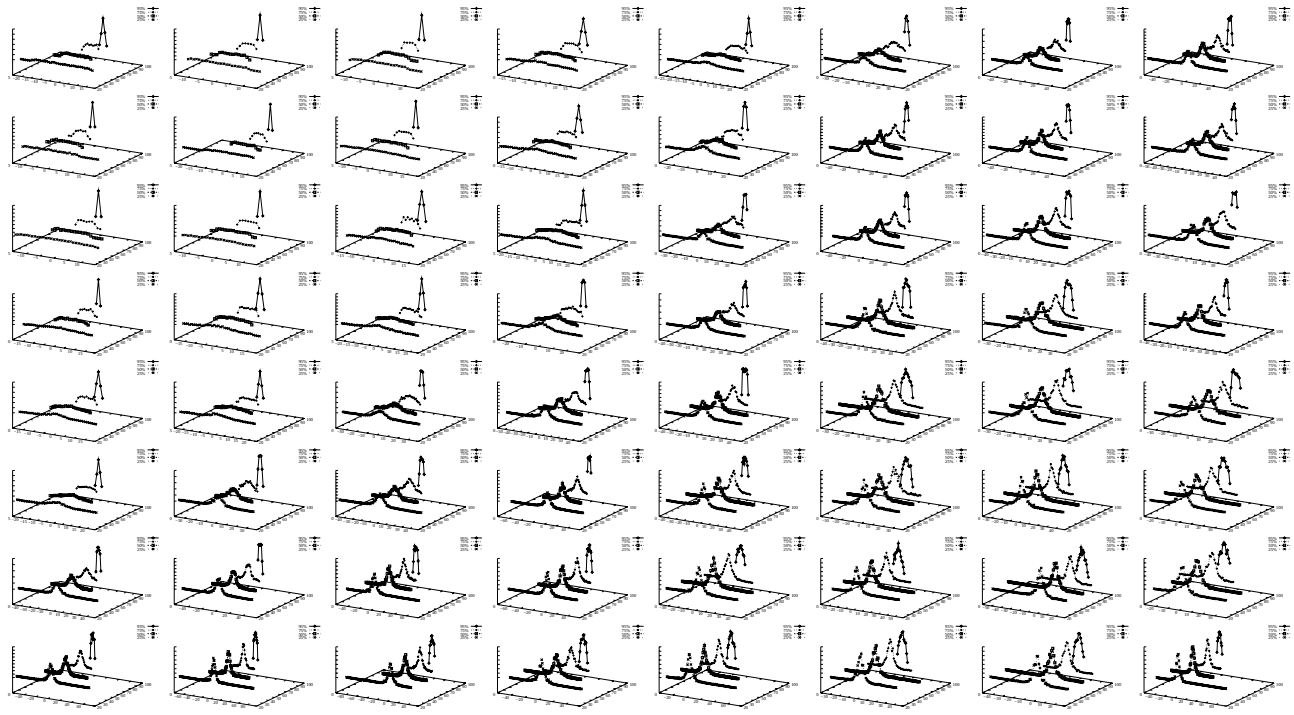


Figure 7.14: Distribution of errors : `peppers`.

7.4.7 Concluding Remarks

The scheme proposed in this section uses the secret linear combination of DCT coefficients so that the cost of finding the secret increases and the addition of the same value to the DCT coefficients results in the different value without knowing the combination. We showed the model of the attacks which use the verification system as the verification oracle. In this model, it is necessary for the attacker to find all groups of blocks to succeed. With the method in which each DCT coefficient belongs to more than one group, the cost of attacking the system will largely increase.

7.5 Conclusion

The original scheme proposed by C. Lin and S. Chang is not secure when the pairing function is known. The scheme does not provide security with the bounding of the DCT coefficient ranges because the difference of two DCT coefficients does not change if the same amount of modification is made to the both coefficients. Such modification does not necessarily produce artifacts and so the modification can be visually undetectable. Our scheme has improvements over SARI system for the following two points : *i*) the cost of finding groups of blocks is largely increased, and *ii*) the MAC size will be smaller

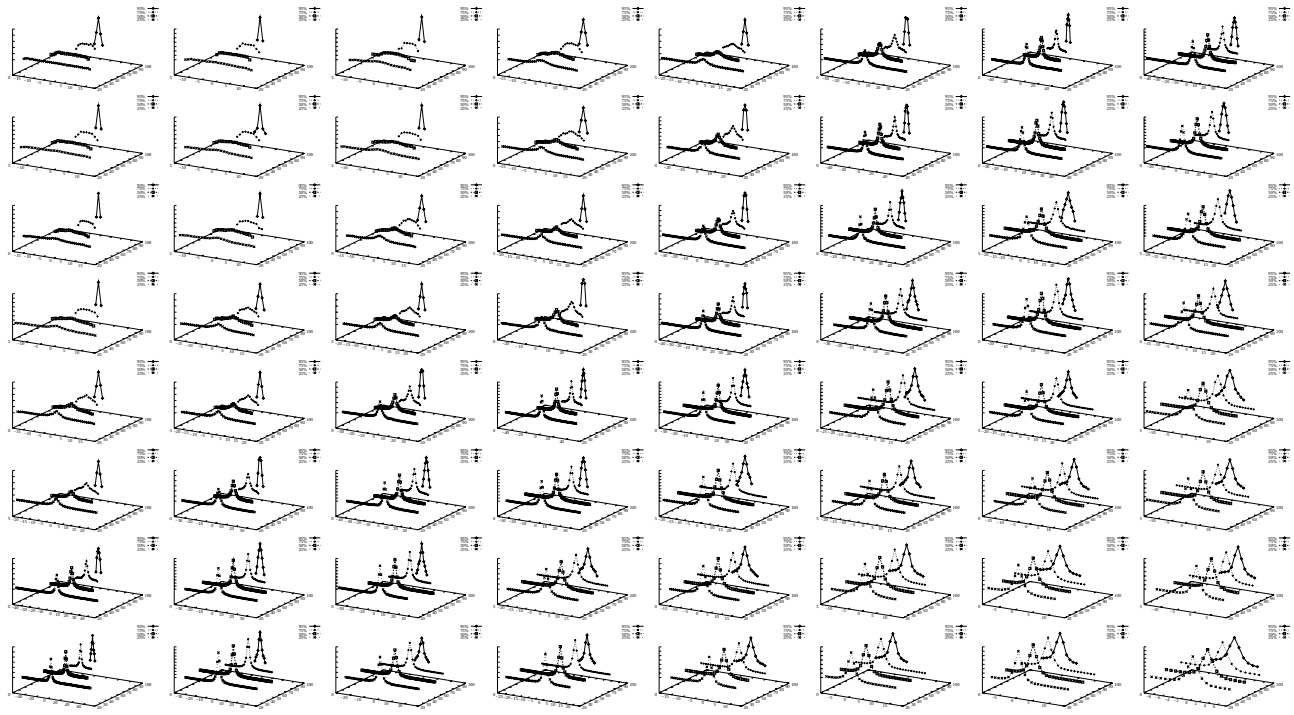


Figure 7.15: Distribution of errors : `airplane`.

for the same level of protection because the sum of more than two blocks is used.

The problem of the security assessment is that there is no established quantitative method to distinguish between the change of an image due to the compression and the malicious modification of an image. The quantitative measure is necessary for the correct security assessment and for this, further research is needed.

Table 7.4: Detection of lena's beauty mark (top) and detection of lena modified by a median filter with 3×3, 5×5, 7×7 and 9×9 window sizes (bottom).

m	Quality	Verification	# of grps failed
8	95%	false	3
	75%	false	3
	50%	false	2
16	95%	false	1
	75%	false	4
	50%	false	1
32	95%	true	
	75%	true	
	50%	true	
64	95%	false	1
	75%	true	
	50%	true	

m	filter size	Verification	# of grps failed
8	3	false	430
	5	false	3462
	7	false	5258
	9	false	6128
16	3	false	228
	5	false	2041
	7	false	3115
	9	false	3524
32	3	false	135
	5	false	1166
	7	false	1742
	9	false	1951
64	3	false	105
	5	false	727
	7	false	971
	9	false	1050

Table 7.5: Tolerance values for linear sums of $m = 8$ (left) and $m = 16$ (right).

33	22	19	33	38	43	41	39
26	25	29	26	37	58	43	55
30	26	28	50	42	35	55	28
19	36	26	31	51	44	25	25
24	23	39	51	49	24	21	32
26	34	39	33	39	23	24	19
30	33	47	19	32	25	24	21
23	29	20	22	23	21	20	12

29	26	19	34	40	47	52	49
21	25	38	37	53	93	60	54
29	29	33	50	63	58	51	35
40	35	41	52	73	80	47	44
29	45	74	91	62	44	54	33
50	37	56	52	34	44	34	29
43	39	32	27	32	25	36	22
29	21	37	22	22	31	21	21

Table 7.6: Tolerance values for linear sums of $m = 32$ (left) and $m = 64$ (right).

47	26	32	50	59	111	84	63	66	40	47	86	77	75	83	86
27	31	30	39	71	92	122	57	60	36	28	56	92	113	77	82
42	38	39	66	53	58	67	58	39	60	40	98	73	90	73	71
42	42	45	58	59	87	66	40	39	58	57	91	144	69	80	43
55	45	80	70	70	77	40	49	66	63	114	113	73	76	51	43
66	57	56	58	50	53	26	54	57	50	70	57	53	56	34	60
45	39	33	54	40	34	26	24	45	63	50	40	39	29	29	32
36	29	22	30	33	33	44	24	46	38	36	33	41	24	34	29

Table 7.7: Tolerance values for linear sums ($m = 128$).

57	47	45	91	83	119	150	67
56	47	69	64	68	125	118	91
57	61	72	113	97	135	112	86
70	85	162	136	97	99	62	54
68	136	110	171	103	76	57	42
99	88	107	99	76	96	64	56
65	44	75	49	71	48	38	30
47	47	68	54	30	37	52	29

Table 7.8: Detection of lena with an 8×8 block at (264,272) position, modified by a median filter with 3×3, 5×5, 7×7 and 9×9 window sizes.

# of coef / group	filter size	Verification	# of groups failed
8	3	true	
	5	false	4
	7	false	8
	9	false	8
16	3	true	
	5	false	4
	7	false	4
	9	false	6
32	3	true	
	5	true	
	7	false	4
	9	false	4
64	3	true	
	5	true	
	7	false	2
	9	false	2
128	3	true	
	5	true	
	7	true	
	9	false	2

Chapter 8

Conclusion

8.1 Introduction

In this thesis, we investigated two security goals for image data, *i*) image encryption that hides the content of images, and *ii*) image authentication that provides assurance for the authenticity of images. We studied existing image encryption and authentication systems and demonstrated various attacks. We proposed a number of security systems and analyzed their security. In this chapter, we summarize image encryption and authentication systems and give some final remarks.

8.2 Image Encryption

To design an encryption system for image data, it is important to understand how a compression system exploits the properties of image data to remove redundancy. In the following paragraphs we summarize the properties used to compress data in the JPEG, MPEG and JPEG2000 compression systems.

In the JPEG and MPEG systems, pixels are transformed into coefficients using the Discrete Cosine Transform. As the result of the transformation, the energy is packed in the lower frequency parts, that is, lower frequency coefficients have larger values and higher ones have smaller values. After quantization of the coefficients, many of the higher frequency coefficients will be zero. JPEG and MPEG exploit this property for compression by using a zig-zag scan and run-length coding of zero coefficients. If encryption changes the order of coefficients in a block, it will result in shorter run-length of zero coefficients and so the compression rate drops.

In JPEG2000, an image is decomposed into different frequency components, i.e. subbands, using the Discrete Wavelet Transform. A subband that consists of wavelet coefficients is divided into code-blocks and each code-block is independently encoded

from other code-blocks. In encoding a code-block, coefficients are divided into bit-planes and bit-planes are encoded one by one in the order of their significance, i.e. from the most significant bit-plane to the least significant bit-plane. To generate *decision-context* pairs, the encoder exploits the correlation of 3×3 bit neighboring regions. Also groups of four consecutive bits are run-length coded to generate *decision-context* pairs. The *decision-context* pairs are encoded using the adaptive binary arithmetic coder. If encryption destroys the correlation of neighboring regions or that of the four consecutive bits, the compression rate will drop.

Image encryption systems must be computationally inexpensive to be able to cope with the large size of image data. To encrypt images, there are two approaches : *i*) using elementary cryptographic operations, and *ii*) selective encryption. These two can be combined together. First we summarize systems using elementary cryptographic operations and then selective encryption systems.

8.2.1 Encryption Using Elementary Cryptographic Operations

Elementary cryptographic operations require small computational power and hence the drop in coding speed can be ignored. In some existing systems, including the JPEG2000 encryption system proposed in this thesis, encryption takes place after quantization and before entropy encoding. Permutation of DCT coefficients as used by Tang [107] and Shin *et al.* [95] is ineffective because of the following reasons.

1. The permutation changes the order of frequencies, and so the higher frequency parts in 8×8 blocks will include many non-zero coefficients. This will result in shorter run-lengths of zero coefficients and so the compression rate will drop.

2. Lower frequency coefficients have larger values and so images can be reasonably approximated by sorting coefficients in blocks. This can be seen in Figure 3.2 and 3.3 in Chapter 3. Hence, the permutation of DCT coefficients by itself does not provide high security.

In our JPEG2000 encryption system, groups of four consecutive bits in a bit-plane are randomly scanned. The random scan was designed to satisfy the following two conditions :

1. The original four bit sequences are kept intact and so the correlation of the four bits is not destroyed.

2. We chose a random scan instead of a random permutation of coefficients because the random scan does not change the positions of coefficients in the code-block and so there is no impact on the correlation of neighboring 3×3 bit regions.

3. The random scan changes the order of *decision-context* pairs that are encoded using the adaptive binary arithmetic coder. However, the adaptive binary arithmetic coder uses an order-0 model and the order of input symbols has very small impact on its compression ratio.

As shown in our experiments in Table 6.2 in Chapter 6, the drop in compression rate due to encryption is less than 3%.

The permuted DCT coefficients can be easily recovered by sorting them but this is not true for the DWT coefficients. This is because in the case of DWT, the coefficients in the same subband (frequency) are permuted while in the case of DCT, those in different frequencies are permuted. If two dimensional discrete wavelet transform is used, DWT coefficients in a subband and pixels at the corresponding region in the image have relationship and so the values of coefficients vary with images. This means that if the original image is not known, it is difficult to recover the correct order of the coefficients.

We showed that the chosen coefficients attack is not effective against the JPEG2000 encryption system. The chosen coefficients attack tries to find the permutation by comparing the original coefficients with the inverse-permuted ones. To obtain the inverse-permuted coefficients, it is necessary to correctly decode the encrypted stream. In JPEG2000, the adaptive arithmetic coder uses more than one model and the order that the models are used is hidden by the random scan. If the same adaptive model is not used in the encoder and the decoder, decoding will fail and so will the attack.

8.2.2 Selective Encryption

Selective encryption allows the use of well-studied security algorithms for encryption. In a selective encryption system, data is encrypted *i*) after quantization and before entropy coding, or *ii*) after entropy coding. The data to be encrypted can be *i*) transformed coefficients, and/or *ii*) decoding parameters such as Huffman tables. The chosen data are encrypted using encryption algorithms such as DES. The choice of the part to be encrypted is crucial for security.

The encryption system proposed by Shin *et al.* [95] encrypts the sign bits of the

difference of two DC coefficients using DES, RC4 and RC5, and permutes AC coefficients. The permutation can be found by a known image attack which compares the coefficients of a known picture sequence with the permuted ones, or the chosen coefficient attack. We have shown in Chapter 4 that if the AC coefficients are known, DC coefficients can be recovered by exploiting the smoothness of images. In an image, neighboring pixels are likely to have similar values. If AC coefficients are known, we can construct DC-free 8×8 pixel blocks from AC coefficients and then estimate DC values of blocks by modifying all pixel values in a block by the same amount such that the pixels on the border of the block and its neighboring ones have similar values.

For a JPEG stream, we showed the following four conditions that need to be satisfied for choosing the parts to be encrypted.

Condition 1 Without the encrypted data, it must be difficult to decode the JPEG stream.

Condition 2 It must be difficult to derive the encrypted data from other information in the same JPEG stream.

Condition 3 The encrypted data must be highly dependent on the image and so the corresponding data from similar images will not be useful.

Condition 4 The search space of the encryption key must be large.

For the JPEG system, we chose the Huffman table specification part, which satisfies these four conditions. It is image dependent and the number of possible the Huffman codes is large. It is shown by Fraenkel and Klein [28] that finding Huffman code for an Huffman coded stream which is similar to the JPEG one, is NP-complete and so finding the Huffman code from the entropy coded data in the JPEG stream will be hard.

Typically the size of Huffman table specification data is less than 200 bytes per JPEG stream and so the additional computational cost for encryption is very small. Since the data is encrypted after compression, there is no impact on compression rate. The typical size for MPEG encryption systems [69, 79, 92, 94] is 10% to 50% of the MPEG stream, although it cannot be directly compared with that of JPEG encryption systems because of the difference of the structures of the two streams.

8.3 Image Authentication

In general image data is in a compressed form due to its large size, and the compression algorithms are usually lossy. Because of this, unlike data authentication systems that must detect a single bit change in data, image authentication systems must remain tolerant to changes due to lossy compression.

Image authentication systems are divided into two classes :

1. Watermarking systems which embed a watermarking signal when signing images and extract the same signal when verifying images.

2. Signature systems which generate hashes from images when signing images and compare the hashes of the original images with the ones generated from the received images when verifying them.

In watermarking systems for image authentication, fragile watermarking is used which is sensitive to modifications. However, if a fragile watermark is embedded in the pixel domain, it is likely not to survive lossy compression. For a fragile watermark to survive lossy compression, the level of the embedded signal must be increased and this will degrade the quality of the image.

On the other hand, signature systems do not degrade image quality. They use either hash functions or a MAC, and so require appending data to the image. For example, a signature system proposed by Bhattacharjee and Kutter [12] extracts features that are signed by the private key of a public key encryption system, and SARI [58] uses a MAC. Our proposed system is an extension of the SARI system. The original SARI system is not secure when the pairing function is known because the difference of two DCT coefficients does not change if the same amount of modification is made to both the coefficients. Such modification can be made visually undetectable. Our scheme improves the SARI system in terms of : *i*) security because the cost of finding groups of blocks is largely increased, and *ii*) efficiency because the MAC size is smaller for the same level of protection.

8.4 Further Research

8.4.1 Image Encryption

There are combined compression and encryption schemes for entropy coders [120, 56, 40, 62, 113]. Since many image compression systems use entropy coders, such combined

compression and encryption systems can replace the original entropy coders in image compression systems. Secure entropy coders usually have a small computation cost and a small drop in compression rate, and so are suitable to be incorporated in the image compression systems. Although there are attacks against the combined compression and encryption systems [11, 56, 40, 113, 112], all known attacks are chosen plaintext attacks and the security of these systems against other types of attacks is an open problem.

8.4.2 Image Authentication

An attack against watermarking systems that are used in trusted devices, such as digital cameras described by Wu and Liu [123], raises an important question about the applicability of image authentication systems. The problem can be described as follows. An attacker can obtain a modified image which will pass the verification check of the authentication system by taking a picture of a modified image using the trusted device. The picture taken is unlikely to contain the original fragile watermark because of its fragility. The verification system will detect the new watermark inserted by the device but it does not detect the old watermark in the original image and so it fails to detect the modifications of the image.

To avoid this attack, Wu and Liu [123] suggested the use of a pair of robust and fragile watermarks. If an image is modified using the attack, the verification system can detect the two robust watermarks, the one which the original image had and the other that was newly inserted by the device. However, most of the proposed fragile watermarking systems are not able to detect multiple watermarks in images. If the attack is used against image signature systems, a new signature will be generated by the trusted device when the picture of the modified image is taken. The verification system does not have ability to find that the picture is taken from the modified image because the signature is not embedded in the image and so there is no evidence that shows the link between the original image and the new signature corresponding to the modified one. For signature systems, protecting against this type of attacks is an open problem.

Although there are many proposed image authentication systems using watermarks, many of them have not been cryptanalyzed. There are publicly accepted definitions of attack operations for robust watermarking systems, and they are implemented in software such as Stirmark [78], Checkmark [24] and Optimark [1]. However, there are no such well defined operations for image authentication systems and such well defined

attack operations are essential to assess security of the systems. To define such attack operations further research is required.

Bibliography

[1] Optimark. *http://poseidon.csd.auth.gr/optimark/*, 2002.

[2] I. Agi and L. Gong. An Empirical Study of Secure MPEG Video Transmissions. *In Proceedings of the Internet Society Symposium on Network and Distributed Systems Security*, pages 137–144, Feb 1996.

[3] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete Cosine Transform. *IEEE Trans. on Computers*, C-23:90–93, 1974.

[4] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. Image Coding Using Wavelet Transform. *IEEE transactions on image processing*, 1:205–220, Apr 1992.

[5] K. Aoki and H. Lipmaa. Fast Implementations of AES Candidates. *Third AES Candidate Conference*, 2000.

[6] G. R. Arce, L. Xie, and R. F. Gravemen. Approximate message authentication codes. *In Proc. 4rd Annual Fedlab Symp. Advanced Telecommunication/Information Distribution*, Mar 2000.

[7] A.Said and W.A.Pearlman. A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees. *IEEE transactions on circuits and systems for video technology*, 6:243–250, 1992.

[8] H. Beker and F. Piper. *Cipher Systems, The Protection of Communications.* Northwood Publications, 1982.

[9] T. C. Bell, I. H. Witten, and J. G. Cleary. *Text Compression.* Prentice-Hall, 1990.

[10] H. A. Bergen and J. M. Hogan. Data security in a fixed-model arithmetic coding compression algorithm. *Computers and Security*, 11:445–461, 1992.

[11] H. A. Bergen and J. M. Hogan. A chosen plaintext attack on an adaptive arithmetic coding compression algorithm. *Computers and Security*, 12:157–167, 1993.

[12] S. Bhattacharjee and M. Kutter. Compression tolerant image authentication. *in Proc. IEEE Int. Conference in Image Processing, vol. 1*, pages 435–439, Oct 1998.

[13] J. Bradley. *xv*. ftp.cis.upenn.edu, 1994.

[14] H. Cheng and X. Li. On The Application of Image Decomposition to Image Compression and Encryption. *Communication and multimedia Security II*, pages 116–127, 1996.

[15] S. Cheng, P. Litva, and A. Main. Trusting DRM Software. *W3C Workshop on DRM, January 2001 : http://www.w3.org/2000/12/drm-ws/pp/cloakware.html*, 2001.

[16] R. J. Clarke. *Transform Coding of Images*. Academic Press, London, 1985.

[17] J. G. Cleary, S. A. Irvine, and I. Rinsma-Melchert. On the Insecurity of Arithmetic Coding
. *http://www.cs.waikato.ac.nz/~sirvine/*, Sep 1995.

[18] Y. Cohen, M. Landy, and M. Pavel. Hierarchical Coding of Binary Images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pages 284–298, 1985.

[19] C. J. Colbourn and J. H. D. (Eds.). *CRC Handbook of Combinatorial Designs*. FL: CRC Press, 1996.

[20] I. J. Cox, J. Kilian, T. Leighton, and T. Shamoon. Secure Spread Spectrum Watermarking for Multimedia. *NEC Research Institute Technical Reporrt 95-10*, Oct 1995.

[21] I. Daubechies. Orthonormal Bases of Compactly Supported Wavelets. *Comm. Pure Appl. Math*, 41:909–996, 1988.

[22] G. Davis. *Baseline Wavelet Transform Coder Construction Kit*. http://www.cs.dartmouth.edu/~gdavis/ wavelet/wavelet.html, 1997.

[23] S. Devadhar, C. Krumbein, and K. M. Liu. *MPEG Background*. http://bmrc.berkeley.edu/research/mpeg/mpeg_overview.html, 2000.

[24] S. P. et al. Checkmark. *http://watermarking.unige.ch/Checkmark/index.htm*, 2002.

[25] F. Müller. Distribution shape of two-dimensional DCT coefficients of natural images. *Electron. Lett.*, 29:1935–1936, 1999.

[26] N. Farvardin and J. W. Modestino. Optimum quantizer performance for a class of non-Gaussian memoryless sources. *IEEE Trans. Inform. Theory*, 30(3):485–497, 1984.

[27] E. F. Foundation. EFF DES Cracker Project. *http://www.eff.org/descracker.html*, 1999.

[28] A. S. Fraenkel and S. T. Klein. Complexity Aspects of Guessing Prefix Codes. *Algorithmica*, 12:972–976, 1994.

[29] J. Fridrich. Image Watermarking for Tamper Detection. *Proc. IEEE Int. Conf. on Image Proc.*, pages 404–408, Oct 1998.

[30] D. L. Gall. MPEG : A Video Compression Standard for Multimedia Applications. *Communications of the ACM*, 34:46–58, Apr 1991.

[31] R. Gonzalez and R. Woods. *Digital Image Processing.* Addison-Wesley, 1992.

[32] R. F. Gravemen and K. Fu. Approximate message authentication codes. *In Proc. 3rd Annual Fedlab Symp. Advanced Telecommunication/Information Distribution*, Feb 1999.

[33] E. Hamilton. *JPEG File Interchange Format.* 1992.

[34] J. J. Hoy. Declaration of John J. Hoy, Superior Court of the State of California. *Available at http://cryptome.org/dvd-v-521.htm#3*, 1999.

[35] D. Huffman. A Method for the Construction of Minimum Redundancy Codes. *Proc. IRE*, 40:1098–1101, Sept 1952.

[36] Image Power, Inc. and University of British Columbia. *JasPer Version 0.072.* http://www.ece.ubc.ca/ mdadams/jasper/, 2000.

[37] R. S. Inc. RSA Laboratories — Cryptography FAQ — Has DES been broken? *http://www.rsasecurity.com/rsalabs/faq/3-2-2.html*, 2003.

[38] A. N. S. Inst. *ANSI X.3.92 American National Standard for Data Encryption Algorithm*. Amer. Nat. Stand. Inst., 1981.

[39] A. N. S. Institute. ANSI X9.31 : Public Key Cryptography Using Reversible Algorithms for the Financial Services Industry: Part 2: The MDC-2 Hash Algorithm. *American National Standard X9.31-1992*, 1993.

[40] S. A. Irvine. PhD thesis
. *http://www.cs.waikato.ac.nz/~sirvine/*, 1995.

[41] S. A. Irvine, J. G. Cleary, and I. Rinsma-Melchert. The subset sum problem and arithmetic coding
. *http://www.cs.waikato.ac.nz/~sirvine/*, Sep 1995.

[42] ISO/IEC. *MPEG Standard*. http://www.mpeg.org, 1998.

[43] ISO/IEC. *JPEG 2000 Part 1 Final Committee Draft Version 1.0*. ISO/IEC JTC 1/SC29 WG 1, March 2000.

[44] ISO/IEC. *JPEG 2000 Verification Model 8.5*. ISO/IEC JTC 1/SC29 WG 1, September 2000.

[45] ITU. *JPEG Standard : CCITT Recommendation T.81*. International Telecommunication Union, 1993.

[46] J. Johansen. DeCSS. *Available at http://www-2.cs.cmu.edu/ dst/DeCSS/*, 1999.

[47] R. L. Joshi and T. R. Fischer. Comparison of Generalized Gaussian and Laplacian Modeling in DCT Image Coding. *IEEE Signal Processing Letters*, 2(5):81–82, 1995.

[48] D. Kirovski, M. Peinado, and F. A. P. Petitcolas. Digital Rights Management for Digital Cinema. *Security in Imaging: Theory and Applications, International Symposium on Optical Science and Technology*, 2001.

[49] D. E. Knuth. The Art of Computer Programming: Sorting and Searching, volume 3. *Journal of Algorithms*, 1973.

[50] D. E. Knuth. Dynamic Huffman Coding. *Journal of Algorithms*, 6:163–180, 1985.

[51] L. Kraft. A Device for Quantizing, Grouping and Coding Amplitude Modulated Pulses. *M.S. Thesis, Dept. Elec. Eng., MIT, Cambridge, MA*, 1949.

[52] M. Kuribayashi and H. Tanaka. A Watermarking Scheme Based on the Characteristic of Addition among DCT coefficients. *Proceedings of ISW2000*, pages 1–14, 2000.

[53] X. Lai and J. L. Massey. A Proposal for a New Block Encryption Standard. *Advances in Cryptology - EUROCRYPT'90 Proceedings*, pages 390–404, 1991.

[54] E. Lam and J. Goodman. A mathematical analysis of the DCT coefficient distributions for images. *IEEE transactions on image processing*, 9:1661–1666, Oct 2000.

[55] Y. Li, Z. Chen, S.-M. Tan, and R. H. Campbell. Security Enhanced MPEG Player. *In Proceedings of IEEE first International Workshop on Multimedia Software Development (MMSD 96)*, pages 169–175, Mar 1996.

[56] J. Lim, C. Boyd, and E. Dawson. Cryptanalysis of Adaptive Arithmetic Coding Encryption Schemes. *ACISP*, pages 216–227, 1997.

[57] C. Lin and S. Chang. A Robust Image Authentication Method Distinguishing JPEG Compression from Malicious Manipulation. *CU/CTR Technical Report 486-97-19*, Dec 1997.

[58] C.-Y. Lin and S.-F. Chang. Robust Image Authentication Method Surviving JPEG Lossy Compression. *Storage and Retrieval for Image and Video Databases (SPIE)*, pages 296–307, 1998.

[59] C.-Y. Lin and S.-F. Chang. SARI : Self-Authentication-and-Recovery Image Watermarking System. *Proc. of ACM Multimedia 2001*, pages 628–629, 2001.

[60] E. T. Lin and E. J. Delp. A Review Of Fragile Image Watermarks. *Proceedings of the Multimedia and Security Workshop, ACM Multimedia '99*, pages 25–29, Oct 1999.

[61] H. Lipmaa. AES Candidates: A Survey of Implementations. *available at http://www.tcs.hut.fi/ helger/aes/*, 2003.

[62] X. Liu, P. G. Farrell, and C. Boyd. Resisting the Bergen-Hogan Attack on Adaptive Arithmetic Coding. *IMA Conference on Coding and Crypt*, 1998.

[63] B. Macq and J. Quisquater. Cryptology for Digital TV Broadcasting. *Proceedings of the IEEE*, 83:944–957, June 1995.

[64] S. G. Mallat. A Theory for Multiresolution Signal Decomposition: The Wavelet Representation. *IEEE transactions on pattern analysis and machine intelligence*, 11:647–693, Jul 1989.

[65] K. Matsui and K. Tanaka. Video-steganography : How to Secretly Embed a Signature in a Picture. *IMA Intellectual Property Project Proceedings*, pages 187–206, Oct 1994.

[66] M. Matsui. Linear Cryptanalysis Method for DES Cipher. *In Advances in Cryptology - Eurocrypt '93, Proceedings*, 765:386–397, 1994.

[67] N. Memon and P. W. Wong. Protecting Digital Media Content. *Comm. of the ACM*, pages 35–43, July 1998.

[68] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press ISBN 0-8493-8523-7, 1996.

[69] J. Meyer and F. Gadegast. *Security Mechanisms for Multimedia-Data with the Example MPEG-I-Video. Project description of SECMPEG*. Technical University of Berlin, Germany, May 1995.

[70] MPEG Software Simulation Group. *MPEG-2 encoder/decoder*. http://www.mpeg.org/MPEG/MSSG/, 1996.

[71] M. Naor and O. Reingold. On the construction of pseudo-random permutations: Luby-Rackoff revisited. *J. of Cryptology*, pages pp29–66, 1999.

[72] M. Naor and O. Reingold. Constructing Pseudo-Random Permutations with a Prescribed Structure. *J. of Cryptology*, 2001.

[73] National Institute of Standards and Technology. *Advanced Encryption Standard (AES)*. http://csrc.nist.gov/publications/drafts/dfips-AES.pdf, 2001.

[74] NEC Electronics America, Inc. *μPD61132 MPEG2 Decoder available at http://www.necelam.com/digitalav/uPD61132.cfm*. 2003.

[75] N.Memon, S.Shende, and P.Wong. On the security of the Yueng-Mintzer Authentication Watermark. *Final Program and Proceedings of the IS&T PICS 99*, pages 301–306, 1999.

[76] U. of Commerce/National Institute of Standards and Technology. FIPS PUB 186-2 : Digital Signature Standard (DSS). *Federal Information Processing Standards Publication*, 2000.

[77] R. Pasco. *Source Coding Algorithms for Fast Data Compression.* 1976.

[78] F. A. P. Petitcolas. Stirmark. *http://www.cl.cam.ac.uk/~fapp2/watermarking/stirmark/*, 2002.

[79] L. Qiao and K. Nahrstedt. A New Algorithm for MPEG Video Encryption. *In Proceedings of The First International Conference on Imaging Science, Systems, and Technology (CISST'97)*, July 1997.

[80] L. Qiao and K. Nahrstedt. Comparison of MPEG Encryption Algorithms. *International Journal on Computers and Graphics, Special Issue: Data Security in Image Communication and NetWork*, Jan 1998.

[81] L. Qiao, K. Nahrstedt, and M.-C. Tam. Is MPEG Encryption Using Random Lists instead of Zig Zag Order Secure? *In Proceedings of 1997 IEEE International Symposium on Consumer Electronics*, Dec 1997.

[82] R. Radhakrishnan and N. Memon. On the Security of the SARI Image Authentication System. *Proc. of International Conference on Image Processing*, pages 971–974, 2001.

[83] J. Rissanen. Generalized Kraft Inequality and Arithmetic Coding. *IBM J.Res.Devel.*, 20:198–203, 1976.

[84] R. L. Rivest. *The MD5 Message Digest Algorithm.* Internet draft RFC1321, April 1992.

[85] R. L. Rivest. *The RC4 Encryption Algorithm.* RSA Data Security, Inc, Mar 1992.

[86] R. L. Rivest. The RC4 Encryption Algorithm. *Dr Dobb's Journal*, 20:146–148, Jan 1995.

[87] R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM*, 21:120–126, 1978.

[88] D. Salomon. *Data compression : the complete reference 2nd edition.* Springer-Verlag NewYork, Inc. ISBN 0-387-95045-1, 2000.

[89] M. Schneider and S. F. Chang. A Robust Content Based Digital Signature for Image Authentication. 1996.

[90] J. M. Shapiro. An Embedded Wavelet Hierarchical Image Coder. *Intl. Conf. on Acoustics, Speech, and Signal Processing*, 4:657–660, 1992.

[91] J. M. Shapiro. Embedded Image Coding Using Zerotrees of Wavelet Coefficients. *IEEE transactions on signal processing*, pages 3445–3462, 1993.

[92] C. Shi and B. Bhargava. A Fast MPEG Video Encryption Algorithm. *In Proceedings of 6th ACM International Multimedia Conference*, pages 81–88, Sep 1998.

[93] C. Shi and B. Bhargava. An Efficient MPEG Video Encryption Algorithm. *In Proceedings of 17th IEEE Symposium on Reliable Distributed Systems*, pages 381–386, Oct 1998.

[94] C. Shi, S.-Y. Wang, and B. Bhargava. MPEG Video Encryption in Real-time Using Secret Key Cryptography. 1999.

[95] S. U. Shin, K. S. Sim, and K. H. Rhee. A Secrecy Scheme for MPEG Video Data Using the Joining of Compression and Encryption. *ISW'99*, pages 191–201, 1999.

[96] B. Shneier. *Applied Cryptography, Second Edition.* John Wiley & Sons, Inc, ISBN0-471-12845-7, 1996.

[97] E. Shusterman and M. Feder. Image Compression via Improved Quadtree Decomposition. *IEEE Trans. on Image Processing*, pages 207–215, 1994.

[98] I. S. P. Society. Signal Processing Magazine : Digital Watermarking, ISSN 1053-888. Sep 2000.

[99] G. Spanos and T. Maples. Performance Study of a Selective Encryption Scheme for the Security of Networked, Real-time Video. *In Proceedings of 4th International Conference on Computer Communications and Network*, Sep 1995.

[100] F. A. Stevenson. Cryptanalysis of Contents Scrambling System. *http://www.derfrosch.de/*, 1999.

[101] F. A. Stevenson. Successfull attack on CSS algorithm. *Livid-dev*, 1999.

[102] B. Stickney. *MPEG-2 or MJPEG?* Videomedia : http://www.videomedia.com/mpeg.htm, 1993.

[103] D. R. Stinson. *Cryptography : Theory and Practice.* CRC Press ISBN 0-8493-8521-0, 1995.

[104] G. Strang. The Discrete Cosine Transform. *G. Strang, The Discrete Cosine Transform, SIAM Review*, 1999.

[105] P. Strobach. Quadtree-structured Recursive Plane Decomposition Coding of Images. *IEEE Trans. on Signal Processing*, pages 1380–1397, 1991.

[106] M. D. Swanson, M. Kobayashi, and A. H. Tewfik. Multimedia Data-Embedding and Watermarking Technologies. *Proc. of the IEEE*, 86:1064–1087, June 1998.

[107] L. Tang. Methods for Encrypting and Decrypting MPEG Video Data Efficiently. *In Proceedings of the ACM Multimedia96*, pages 219–229, Nov 1996.

[108] D. Taubman. High Performance Scalable Image Compression with EBCOT. *Proceedings of International Conference on Image Processing*, 3:344–348, 1999.

[109] D. Taubman. High Performance Scalable Image Compression with EBCOT. *IEEE Transactions on Image Processing*, 9:1158–1170, July 2000.

[110] D. Taubman and A. Zakhor. Multirate 3-D Subband Coding of Video. *IEEE transactions on image processing*, 3:572–588, Sep 1994.

[111] The Independent JPEG Group. *JPEG software release 6b (jpeg-6b).* http://www.ijg.org, 1998.

[112] T. Uehara and R. Safavi-Naini. Attack on Liu/Farrell/Boyd Arithmetic Coding Encryption Scheme. *Proc. of Communications and Multimedia Security Joint Work Conference IFIP TC6 and TC11*, Sep 1999.

[113] T. Uehara and R. Safavi-Naini. Attacking and Mending Arithmetic Coding Encryption Schemes. *Proc. of Australasian Computer Science Conference*, pages 408–419, Jan 1999.

[114] T. Uehara and R. Safavi-Naini. Chosen DCT Coefficients Attack on MPEG Encryption Schemes. *Proc. of IEEE Pacific-Rim Conf. on Multimedia*, pages 316–319, Dec 2000.

[115] B. University of California. *MPEG Background.* http://bmrc.berkeley.edu/research/mpeg/, 2000.

[116] R. Venkatesan, S.-M. Koon, M. H. Jakubowski, and P. Moulin. Robust Image Hashing. *Proc. IEEE Int. Conf. on Image Proc.*, 2000.

[117] S. Verdu'. Fifty Years of Shannon Theory. *IEEE Trans. on Information Theory*, pages 2057–2078, Oct 1998.

[118] J. D. Villasenor, B. Belzer, and J. Liao. Wavelet Filter Evaluation for Image Compression. *IEEE transactions on image processing*, 4:1053–1060, Aug 1995.

[119] E. W. Weisstein. Steiner System – from MathWorld. *MathWorld : http://mathworld.wolfram.com/SteinerSystem.html*, 1999.

[120] I. H. Witten and J. G. Cleary. On the Privacy Afforded by Adaptive Text Compression. *Computers and Security*, 7:397–408, 1988.

[121] I. H. Witten, R. Neal, and J. G. Cleary. Arithmetic Coding for Data Compression. *Comm ACM*, pages 520–540, June 1987.

[122] P. W. Wong. A Public Key Watermark for Image Verification and Authentication. *Proc. IEEE Int. Conf. on Image Proc.*, 1:455–459, Oct 1998.

[123] M. Wu and B. Liu. Attacks on Digital Watermarks. *33th Asilomar Conference on Signals, Systems, and Computers*, pages 1508–1512, 1999.

[124] W. Wu and B. Liu. Watermarking For Image Authentication. *Proc. IEEE Int. Conf. on Image Proc.*, pages 437–441, 1998.

[125] L. Xie and G. R. Arce. Joint wavelet compression and autheitcation watermarking. *In Proc. IEEE Int. Conf. on Image Processing*, pages 427–431, Oct 1998.

[126] L. Xie and G. R. Arce. Approximate Image Message Authentication Codes. *IEEE Trans. on Multimedia*, pages 242–252, June 2001.

[127] M. Yeung and F. Mintzer. An Invisible Watermarking Technique for Image Verification. *Proc. IEEE Int. Conf. on Image Proc.*, 1997.

[128] M. M. Yeung. Digital Watermarking. *Comm. of the ACM*, pages 31–33, July 1998.

[129] E. Young. *DES library*. FreeBSD, 1997.

[130] J. Ziv and A. Lempel. A Universal Algorithm for Sequential Data Compression. *IEEE Transactions on Information Theory*, IT-23:337–343, May 1977.

[131] J. Ziv and A. Lempel. Compression of Individual Sequences via Variable-rate Coding. *IEEE Transactions on Information Theory*, IT-24:530–536, Sept 1978.