DIGITAL RIGHTS MANAGEMENT (DRM) - WATERMARK ENCODING SCHEME FOR JPEG IMAGES

by

Sindhu Samuel
(20030500)

Submitted in partial fulfillment of the requirements for the degree
Master of Engineering (Computer Engineering)
in the
Faculty of Engineering, the Built Environment and Information Technology
UNIVERSITY OF PRETORIA

August 2007

SUMMARY

DIGITAL RIGHTS MANAGEMENT (DRM) - WATERMARK ENCODING SCHEME FOR JPEG IMAGES

by

Sindhu Samuel

Supervisor & Co-Supervisor:

Prof G.P. Hancke & Prof W.T Penzhorn

Department of Electrical, Electronic & Computer Engineering

Master of Engineering (Computer)

The aim of this dissertation is to develop a new algorithm to embed a watermark in JPEG compressed images, using encoding methods. This encompasses the embedding of proprietary information, such as identity and authentication bitstrings, into the compressed material. This watermark encoding scheme involves combining entropy coding with homophonic coding, in order to embed a watermark in a JPEG image. Arithmetic coding was used as the entropy encoder for this scheme.

It is often desired to obtain a robust digital watermarking method that does not distort the digital image, even if this implies that the image is slightly expanded in size before final compression. In this dissertation an algorithm that combines homophonic and arithmetic coding for JPEG images was developed and implemented in software. A detailed analysis of this algorithm is given and the compression (in number of bits) obtained when using the newly developed algorithm (homophonic and arithmetic coding).

This research shows that homophonic coding can be used to embed a watermark in a JPEG image by using the watermark information for the selection of the homophones. The proposed algorithm can thus be viewed as a ‘key-less’ encryption technique, where an external bitstring is used as a ‘key’ and is embedded intrinsically into the message stream.

The algorithm has achieved to create JPEG images with minimal distortion, with Peak Signal

to Noise Ratios (PSNR) of above 35dB. The resulting increase in the entropy of the file is within the expected 2 bits per symbol. This research endeavor consequently provides a unique watermarking technique for images compressed using the JPEG standard.

Keywords: Digital watermarking, Arithmetic coding, Homophonic coding, JPEG, entropy coding, source modeling, 'key-less' encryption technique, cryptography, message randomization, copyright.

SAMEVATTING

DIGITAL RIGHTS MANAGEMENT (DRM) - WATERMARK ENCODING SCHEME FOR JPEG IMAGES

deur

Sindhu Samuel

Studieleiers: Prof G.P. Hancke & Prof W.T. Penzhorn

Departement Elektriese, Elektroniese & Rekenaar Ingenieurswese

Meester in Ingenieurswese (Rekenaar)

Die doel van hierdie verhandeling is om 'n nuwe algoritme te ontwikkel om 'n watermerk in JPEG saamgedrukte beelde te verskans, deur van enkoderingsmetodes gebruik te maak. Dit behels die verskansing van eienaarsinligting, soos die identiteit- en magtigingsbisstringe in die saamgedrukte materiaal. Die watermerk-koderingskema kombineer entropiekodering met homofoniese kodering, met die doel om 'n watermerk in 'n JPEG beeld te verskans. Rekenkundige kodering is as entropiekodeerder in hierdie skema gebruik.

Die gewenste uitkoms is 'n kragtige watermerkmetode wat nie die digitale beeld sal verwring nie, al impliseer dit dat die beeld voor die finale saamdrukking effens vergroot word. In hierdie verhandeling is 'n algoritme ontwikkel wat beide homofoniese en rekenkundige kodering vir JPEG beelde in sagteware ïmplimenteer. 'n Deeglike ontleding van die algoritme en die saamdrukking (in getal bisse) wat verkry word, wanneer die nuut ontwikkelde algoritme gebruik word (homofoniese en rekenkundige kodering), word verstrek.

Hierdie narvorsing bewys dat homofoniese kodering gebruik kan word om 'n watermerk in 'n JPEG-beeld te verskans, deur die watermerkinligting te gebruik om die homofone te selekteer. Die voorgestelde algoritme kan dus beskou word as 'n 'sleutellose' enkripsietegniek, waar 'n eksterne bisstring as 'n 'sleutel' gebruik word en intrinsiek in die boodskapstroom verskans word.

Die algoritme het daarin geslaag om JPEG-beelde met minimale verwringing t.o.v. Piek-Sein-tot-Ruis-Verhouding (PSRV) van groter as 35dB daar te stel. Dit het tot gevolg dat die

entropie van lêers tot binne die verwagte twee bisse per simbool toeneem. Hierdie navorsingspoging voorsien dus ŉ unieke watermerkingstegniek vir beelde wat saamgedruk word deur die JPEG-standaard te gebruik.

Sleutelwoorde: Digitale watermerking, rekenkundige kodering, homofoniese kodering, JPEG, entropiekodering, bronmodellering, 'sleutellose' enkripsietegniek, kriptografie, boodskap-ewekansigmaking, outeursreg.

ACKNOWLEDGEMENT

Embarking on this journey and exploration of a new and exciting field has certainly been an exclusive experience of unique caliber. On this journey, there have been many moments and memories for which acknowledgement is required and hence I seize this opportunity.

I would like to express my gratitude to my supervisors, Prof Hancke and Prof Penzhorn for all the guidance, advice and willingness to support my progress in this dissertation. It truly has been an incredible opportunity and special mention and appreciation must be made to Prof Penzhorn who spurred this dissertation and encouraged me to explore the fascinating domain of Information security.

There is heartfelt appreciation to the many friends and colleagues who have encouraged me and provided their help, friendship, understanding and support during this dissertation.

From the very beginning to the very end of this educational journey, I am absolutely indebted to my entire family especially Daddy & Mummy for their continual prayers, encouragement and inspiration to bring this dissertation to completion. Had it not been for your love & support, this dissertation would not have come to fruition.

As with any journey, God has guided and provided every step of the way. Without Him, existence and exploration would not have any meaning or fulfilment. May He continue securing the future whether it be of the Information we daily work with or any other matter on this earth.

TABLE OF CONTENTS

PAGE

LIST OF ABBREVIATIONS

BMP	Bitmap
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DRM	Digital Rights Management
DWT	Discrete Wavelet Transform
FDCT	Forward Discrete Cosine Transform
FFT	Fast Fourier Transform
GIF	Graphics Interchange Format
IDCT	Inverse Discrete Cosine Transform
JPEG	Joint Photographic Experts Group
PSNR	Peak Signal to Noise Ratio
RLE	Run Length Encoding

CHAPTER 1. INTRODUCTION

1.1. BACKGROUND AND PROBLEM STATEMENT

In this present era of information technology and rapid technological advancements, the Internet is increasingly playing a role in the offering of multimedia resources through digital networks. This poses a threat of unauthorized possession and usage of digital material. An additional threat is the illegal tampering and modification of digital material. Digital multimedia has the disadvantage of being prone to easy illegal copying methods such as piracy, counterfeiting and fraud. The need for techniques to protect the copyright of digital material thus arises. The present challenges and factors that need to be taken into consideration, when searching for such a technique, are the protection of ownership rights against illegal activities in such a manner that the ownership of the digital material is beyond dispute. This is particularly pertinent to visual media such as JPEG and MPEG material since these are two of the most popular forms of visual digital material on the Internet.

Digital Rights Management (DRM) aims to address the protection of digital content throughout its lifetime and is currently becoming increasingly important in the protection of digital material made available on the Internet. This includes digital material such as computer programs, music and images in digital format as well as multi-media material. In these areas Digital Rights Management includes aspects such as copyright protection, verification of ownership, distribution and usage control, identification and verification of legitimate users, prevention/detection of illegal copying, etc. The development of secure content delivery systems that incorporate Digital Rights Management is currently drawing a significant amount of interest in industry. The main DRM technology that this dissertation focuses on is that of digital watermarking.

In many cases the digital material is compressed according to the Joint Photographic Expert Group (JPEG) standard. The essential challenge addressed here is to provide a digital

watermarking technique for JPEG images based on encoding techniques, as an enhancement over spectral watermarking techniques, so as to achieve no distortion of the image once the watermark is inserted.

1.2. AN OVERVIEW OF DIGITAL WATERMARKING

Living in a time where information networks are evolving and changing at such a rapid pace, intellectual copyright protection is becoming increasingly important. This is due to digital data being particularly simple to copy and resell without any loss of quality [1]. Digital representation and distribution of data has increased the potential for misuse and theft and thus giving rise to problems associated with copyright protection and the enforcement of these rights. In recent research on security threats to Electronic commerce according to [2], it has been found that another main contributor to the above mentioned problem is the fact that people are ignorant or unaware of copyright restrictions that protect intellectual property.

Intellectual property is the ownership of ideas and control over the tangible or virtual representation of these ideas. The copyright infringement of these ownership rights causes damage in some way or another usually monetary damage to the copyright holder. The enforcement of copyrights on the Internet such as perceiving when a digital image has been borrowed, cropped or illegally used on a web page is an extremely difficult task. The challenge is thus to display and to make digital products available on the Internet while simultaneously protecting these copyrighted works.

There are basically two main technical approaches to address the challenge of intellectual copyright protection namely usage control and digital watermarking techniques [1]. The most promising technique was found to be that of digital watermarking.

1.2.1. Digital watermarking and types of watermarks

The main aspiration of digital watermarking is to protect the intellectual property of

multimedia contents. The digitization of our world has expanded the concept of watermarking in order to be used in authenticating ownership claims and protecting proprietary interests. Digital watermarking can be applied to various different digital products such as images, audio, video, text, graphics and certificates. The focus here will be on implementing watermarking for the protection of JPEG images. Thus 'watermarking' is referred to in the context of digital images for the remainder of this document.

Definition of a watermark: A digital watermark is a signal that is embedded in digital material in order to allow one to establish ownership, identify a buyer or provide some sort of copyright information. As discussed in [1] this may deter people from illegal copying by allowing the determination of the legitimate owner of the protected image and the corresponding copyright. Digital watermarks serve an important role in providing evidence for copyright infringements and thus making the misuse of protected images traceable.

There are two main categories that a watermark can fall under, namely visible and invisible watermarks. Visible watermarks are designed to be perceptible to the user and typically contain a visual message or company logo that clearly indicates the ownership of the image. Most research currently focuses on invisible watermarks which are imperceptible under normal viewing circumstances. The objective of both categories of watermarks is to permanently and unalterably mark the image so that the credit or assignment is beyond dispute.

Visible watermarks are especially useful for conveying an immediate claim of ownership. The main advantage of visible watermarks, in principle at least, is that they virtually eliminate the commercial value of the document to a 'potential' thief without lessening the document's utility for legitimate and authorized purposes [3].

An invisible watermark is intended to be imperceptible but is detected and extracted by an appropriate piece of software when the need arises. An image containing an invisible watermark should look similar to the original unmarked image. It is required that the

watermarked image should suffer no perceptible quality degradation from the original. These watermarks must also be non-repudiable in that it must be verifiable to anybody that they are embedded and what they mean. This type of watermark is useful as a means of identifying the source, author, creator, owner and authorized consumer of a document or image. In the event of illicit usage, the watermark would facilitate the claim of ownership, the receipt of copyright revenues or the success of prosecution [3]. Invisible watermarks play a role in catching the thief rather than discouraging the theft in the first place.

A watermark can be classified as fragile, semi-fragile or robust. Fragile watermarks are corrupted or distorted by any image processing transformation. Semi-fragile watermarks are designed to become corrupt when subjected to any change that exceeds a user-specified threshold. Thus a threshold of zero would form a fragile watermark. Robust watermarks withstand common image processing operations such as rescaling, cropping, compression, etc. This implies that the watermark should still remain intact and should be detectable and recovered despite the signal processing attacks.

1.2.2. Digital watermarking techniques

The specific requirements of each watermarking technique may vary with the application and there is no universal watermarking technique that satisfies all the requirements for all applications. There are several techniques for embedding information (data signal or pattern), more commonly referred to as the watermark, into a digital image. Some of the main technique categories used to solve the problem of copyright protection are discussed below.

Spatial watermarks are created in the image spatial domain and are embedded directly into an image's pixel data. Spectral or transform based watermarks are embedded into an image's transform coefficients (DCT, Wavelet). Blind watermarking methods perform verification of the watermark without the use of the original image. Most other techniques need the original image in order to detect the watermark. Image-adaptive watermarking techniques are usually transform-based and are very robust. These were developed for image compression. A detailed

description of the most pertinent techniques is given in Chapter 2 of this dissertation.

1.3. OBJECTIVES

This dissertation focuses on digital material that has been compressed according to the JPEG standard and an important aspect of this standard is that the coefficients are compressed by using entropy encoding such as Huffman or Arithmetic coding. In these entropy coding methods binary code words are assigned according to the probability of occurrence of the individual coefficients, giving data compression. This applies to certain watermarking techniques that are used to provide copyright protection for digital images. Homophonic coding is a well-established cryptographic coding technique that exploits the probabilistic properties of a message source [4]. Homophonic coding provides data compression whilst at the same time introduces external random information into the encoded message stream as part of the inherent coding process. The external information may be utilized to embed proprietary information into the message stream, by using the probability distribution of the individual message source symbols.

It is often desired to obtain a robust digital watermarking method that does not distort the digital image in any manner even if this implies that the image is slightly expanded in size. Thus the main goal of this dissertation is to design and implement an algorithm that combines homophonic and arithmetic coding for the watermarking of JPEG images. This research endeavor consequently aims to provide a unique and robust watermarking technique.

1.4. RESEARCH APPROACH

1.4.1. Research Methodology

This section includes a depiction of the methodology followed. Figure 1.1 depicts the implementation overview and the sequence followed in researching each component of the newly developed algorithm and thus implementing each component.

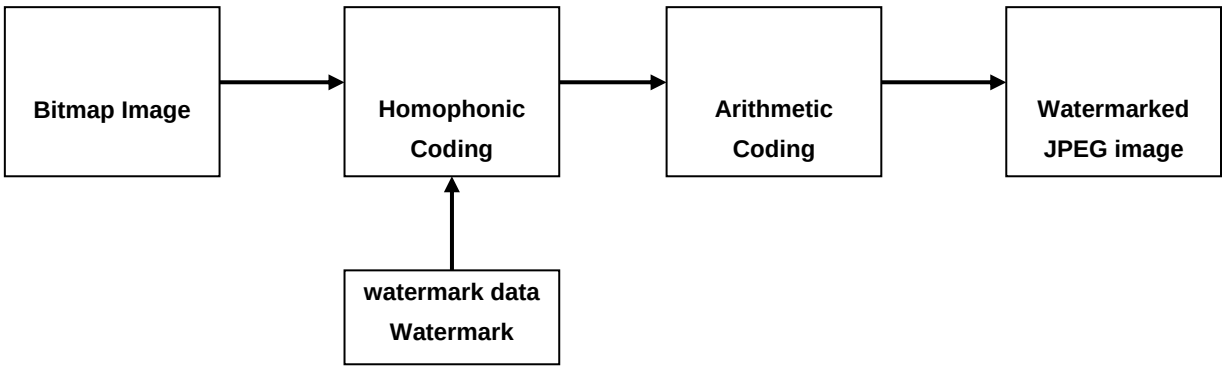

Figure 1.1 JPEG watermark encoding scheme

The first step was to investigate the JPEG standard and explore the IJG library. Following which a detailed investigation of the arithmetic encoding scheme was performed. This led to the implementation of arithmetic coding for JPEG compressed images. Homophonic coding was then explored and the integration of homophonic coding and arithmetic coding for JPEG images. The implementation of Homophonic coding making use of the watermark data & combining with Arithmetic coding for JPEG was performed as per the following depicted approach.

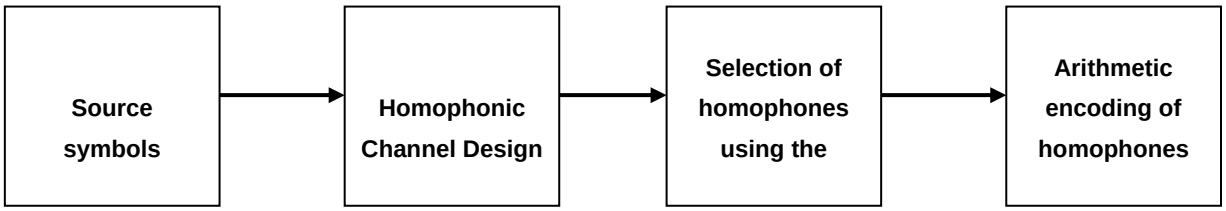

Figure 1.2 Homophonic coding based on arithmetic encoding

This provides the essence of the methodology followed for the research and implementation of this dissertation.

1.4.2. Research Contribution

The contributions that will be made by this dissertation:

- An investigation of the current digital watermarking techniques with respect to JPEG images, an in depth exploration of the JPEG compression standard, Homophonic coding as well as Arithmetic coding.
- The development of a new algorithm that will allow the embedding of proprietary information, such as identity and authentication bitstrings, into compressed material in specific into JPEG images.
- The underlying idea of the proposed algorithm is to combine entropy coding (Arithmetic coding) with homophonic coding while embedding a watermark in the process such that minimal or no distortion is introduced into the image once the watermark is inserted.
- An analysis of the compression (in number of bits) obtained when using the newly developed algorithm (homophonic and arithmetic coding) and a comparison of this with normal JPEG compression.
- An evaluation of the security aspects of the newly developed ('Homophonic encoding') algorithm.

1.5. RESEARCH OUTLINE

This dissertation consists of several chapters in which the various aspects of the new digital watermarking technique are explored and discussed.

Chapter 2 presents a study of the existing watermarking techniques and the security aspects thereof. This mainly focuses on watermarking techniques that have been developed for compressed digital material.

Having covered the currently existing watermarking techniques, Chapter 3 covers the JPEG compression standard and will provide a detailed description of the encoding scheme used and

the source probability estimation techniques.

Chapter 4 involves an in-depth exploration of Homophonic and Arithmetic coding and examples of how these two coding schemes work is also given.

Chapter 5 will engage a thorough discussion of the design and implementation of the new algorithm for the unique watermarking of JPEG images. This includes descriptions of the homophonic method implemented on the probabilities estimated from a JPEG image and the technique used to include proprietary data. This also involves an in-depth explanation of the operation of the newly developed algorithm and the significant achievements with respect to obtaining a robust watermarking technique.

Chapter 6 discusses the security aspects as well as the compression aspects of the newly developed algorithm.

Chapter 7 concludes with a summary of the research performed and what has been achieved in this dissertation. It will also include suggestions for further work and future research based on this topic.

CHAPTER 2. DIGITAL WATERMARKING TECHNIQUES

2.1. ORIGIN OF WATERMARKING

Digital watermarking is the process that embeds digital data called a watermark into a multimedia object such that the watermark can be detected or extracted later to make an assertion about the object [5]. Watermarking seeks to embed a unique piece of data into the original work.

Digital watermarking techniques ensure that some copyright-related information is available to anyone who is interested in the legitimate ownership of the intellectual property. In this particular dissertation the intellectual property which is the centre of concern is that of JPEG images.

2.2. CHARACTERISTICS OF DIGITAL WATERMARKS

Watermarks are intended to be permanently embedded in the host data, in this case the original image. In the event where the owner of the image is in question, the information extracted from the watermark should verify the owner. In order to be effective in the protection of the ownership of intellectual property, a watermark needs to meet certain requirements. These requirements are listed below [6]:

- **Imperceptible:** The watermark should be perceptually invisible such that the experience of viewing the image is not affected.
- **Robustness:** The watermark must survive image modifications that are common to typical image-processing applications e.g. scaling, cropping and image compression to name a few. This implies that the watermark should still be retrievable even after

common signal processing operations such as geometric image operations and noise filtering have been applied.

- **Undeletable:** The watermark should be difficult to remove by an unauthorized user, without degrading the original image.
- **Unambiguous:** The recovery of the watermark should unmistakably identify the owner of the watermarked image.

2.3. CLASSIFICATION OF WATERMARKING TECHNIQUES

As mentioned previously, the specific requirements of each watermarking technique may vary with the application and there is no universal watermarking technique that satisfies all the requirements completely for all applications. In general, any watermarking technique involves an embedding and a detection process. An overview of a general watermarking method is given in Figure 2.1 below.

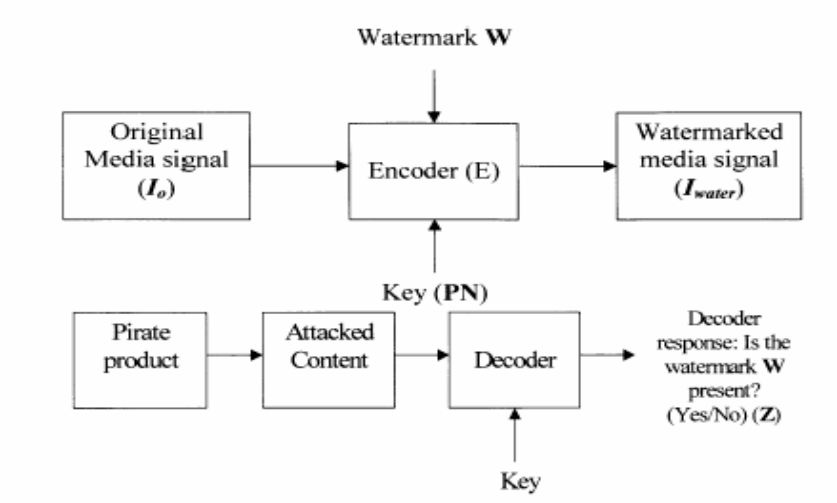

Figure 2.1 A generic watermarking system, including embedding and detection. [8].

The embedding process typically consists of watermark generation, which usually includes the encryption of the watermark and the embedding of the watermark thus resulting in a watermarked image. In the watermark embedding process, given the original image *I* and the

watermark *W*, the watermarked image *I'* can be expressed mathematically by the following equation.

$$I' = I + f(I,W) \tag{2.1}$$

The original image *I* and the watermark *W* serve as inputs of the system and an optional public or secret key K can be used in a watermarking scheme [9]. The output of the system results in a watermarked image *I'*. The watermark detection process establishes the existence of a watermark in the digital content.

There are several techniques for embedding a watermark into a digital image. In general, watermarking techniques for images are classified into two main categories namely the spatial domain approach and the frequency domain approach [3].

The spatial domain approach entails spatial watermarks which are created in the image spatial domain and are embedded directly into an image's pixel data. Frequency domain or spread spectrum techniques involve spectral or transform based watermarks which are embedded into an image's transform coefficients using the Discrete Cosine Transform (DCT), Discrete Wavelet Transform (DWT), Discrete Fourier Transform (DFT) and the Fast Fourier Transform (FFT). These methods work by using the transforms to analyse the data and thus manipulate the coefficients produced by the transforms in order to embed the information into an image [7]. Other transform domain techniques include the Mellin-Fourier transform, Fractal Transform, etc. An overview of the two main categories of watermarking techniques is depicted in Figure 2.2 below.

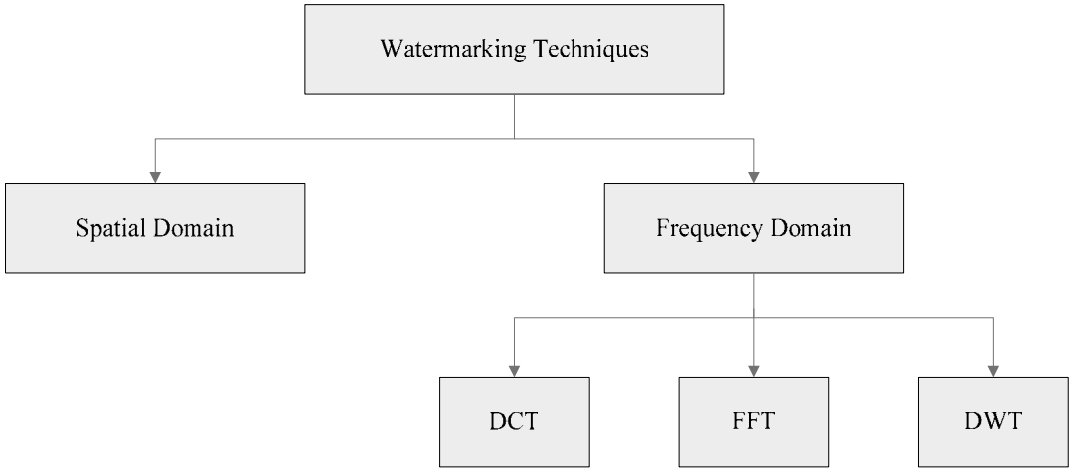

Figure 2.2 Overview of watermarking techniques.

Other significant watermarking techniques are briefly described here. Blind watermarking methods perform verification of the watermark without the use of the original image. Most other techniques need the original image in order to detect the watermark and are usually more robust. Image-adaptive watermarking techniques are usually transform-based and are very robust. Image adaptive techniques were mainly developed for image compression.

Steganographic and non-steganographic watermarking techniques where steganographic watermarking techniques ensures that the users are unaware that a watermark is present and in non-steganographic watermarking, the users are aware that a watermark exists and is thus used to dissuade piracy [6]. There also exists asymmetric and symmetric watermarking where asymmetric watermarking is a technique where two different keys are used for the embedding and detection process. Symmetric watermarking entails the use of the same key for the embedding and detecting of a watermark.

2.3.1. Spatial domain techniques

Spatial domain watermarking methods scatter the data to be embedded to make the data hardly noticeable. Image watermarking in the spatial domain involves modifying pixels in order to embed information. The most common spatial watermarking techniques are the Least

Significant Bit (LSB) technique and the patchwork algorithm [9].

2.3.1.1. The Least Significant Bit technique

The LSB technique allows for the effortless embedding of data in a fixed number of LSBs of the image pixel. This method is based on masking and the watermark is embedded by choosing the 1's and 0's, constituting the watermark data, for the LSBs. This technique was originally designed to work with gray-scale images but is easily extended to colour images by treating each colour plane as a single plane in which data is inserted in the LSBs [6]. The watermark can thus be extracted or decoded if the positions of the LSBs used in the embedding process are known. The main advantage of the LSB technique is that the implementation of such a technique is computationally inexpensive.

2.3.1.2. The Patchwork Algorithm

The patchwork algorithm uses a secret key to initialize a pseudorandom number generator which outputs the locations of the cover which will host the watermark. In the embedding process, n-pixel pairs are randomly chosen according to the secret key. The luminance values (a_i, b_i) of the n pairs of pixels are modified by adding 1 to all the values of a_i and subtracting 1 from every b_i. This results in an increase in brightness of a_i and a decrease in the brightness of b_i. A statistical approach is used for detection where the sum

$$S = \sum_{i=1}^{n} a_i - b_i \tag{2.2}$$

is computed and a_i and b_i in the above equation are the values after they have been modified by 1. If the image contained a watermark then the sum is expected to be 2n else it should be approximately zero. This is based on the assumption that the randomly chosen pixel pairs are independent and identically distributed [10].

It should be kept in mind that any approach that modifies an image using spatial techniques is highly sensitive to noise and can be easily destroyed. Spatial techniques may also cause the image to be degraded by the insertion of the watermark. Spatial domain techniques are strong against cropping and translation but weak to attacks such as noise and compression [9].

2.3.2. Frequency domain techniques

The most widely used methods of data transformation in the frequency domain are DCT, DWT and DFT. In these methods, the watermark to be embedded is distributed over the whole domain of the cover image and this is achieved by applying a transformation to the original image.

2.3.2.1. Discrete Cosine Transform

The Discrete cosine transform was originally widely studied by the source coding community in the context of JPEG and MPEG and was later also considered for the use of embedding a message inside images and videos [10]. This then developed into the use of DCT for watermarking as well. It is stated that DCT is the most widely used transform technique in image coding [11].

An image consists of many pixels arranged in an m x n array. In a DCT-based watermarking scheme [42], the original image is first divided into 8x8 blocks of pixels. The size of the blocks has been chosen as a compromise of complexity and quality. The two-dimensional DCT is then performed independently on each block to obtain the DCT coefficients for each 8x8 pixel block. This results in 64 DCT coefficients for each block. Coefficients of the middle-frequency range are picked from the DCT coefficients and are modified so that their relative values encode a one or zero in order to embed the watermark bits. The middle frequency coefficients are usually chosen due to their moderate variance property (i.e. they have similar magnitudes). This is required so that the modification of DCT coefficients is not perceptually noticeable. The watermarked image is then obtained by performing the inverse

DCT (IDCT) of each block.

The major benefits of DCT include its high energy compaction properties and the availability of fast algorithms for the computation of the transform. The energy compaction property of the DCT results in transform coefficients with only a few coefficients having significant values [11], thus making it well-suited for watermarking. Embedding rules in the DCT domain are often more robust to JPEG and MPEG compression, thus allowing the watermark designer to prevent JPEG/MPEG attacks more easily. Another main advantage of watermarking in the DCT domain is that it provides the possibility of directly realizing the embedding operator in the compressed domain (i.e. inside a JPEG or MPEG encoder) in order to minimize the computation time [10].

A comprehensive description of the DCT is given later on in this dissertation when the JPEG standard is discussed.

2.3.2.2. Discrete Wavelet Transform

The Discrete wavelet transform consists of multiscale frequency decomposition of an image. In the case of a DWT for a two-dimensional image, the image is firstly decomposed into four parts of high, middle and low frequencies (i.e. LL_1, HL_1, LH_1, HH_1) by subsampling horizontally and vertical channels using subband filters [12]. These subbands can be decomposed further to acquire the next coarser scaled wavelet coefficients. Figure 2.3 shows a conceptual example of an image being decomposed with three scale factors.

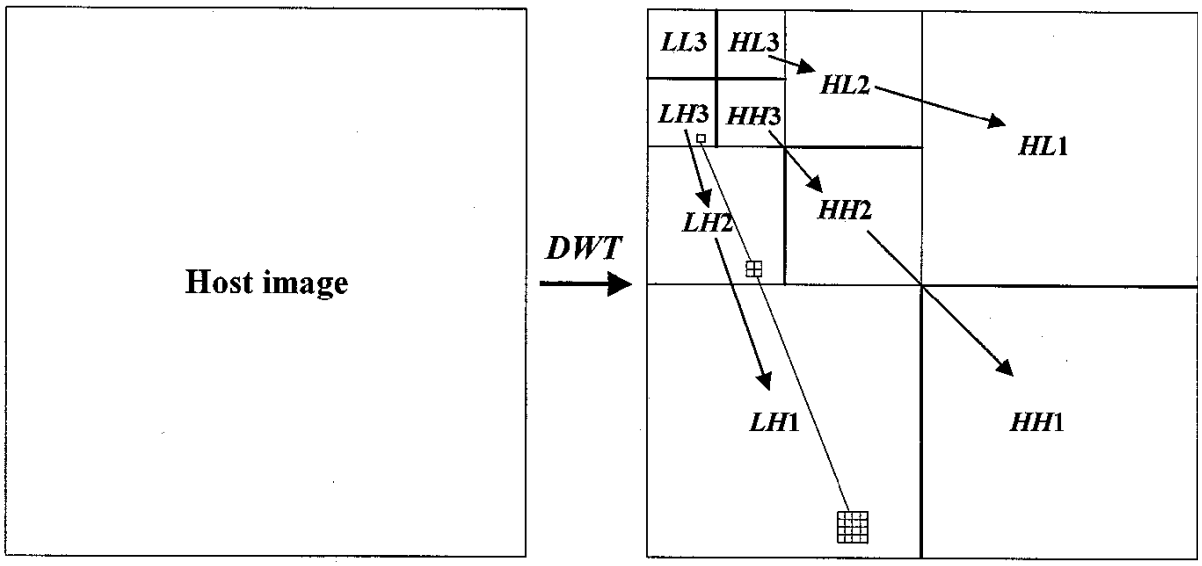

Figure 2.3 Multiscale decomposition of an image [12].

The lowest frequency band at the lowest scale factor, in this case a scale factor of 3 is used, can be found in the top-left corner i.e. LL_3. At the same resolution level, block HL_3 contains information about the highest horizontal and lowest vertical frequency band [10]. LH_3 contains information about the lowest horizontal and the highest vertical frequency band and HH_3 contains information about the highest horizontal and the highest vertical frequency band at the lowest scale factor. The same process is repeated for the intermediate and highest resolution levels. This DWT decomposition is demonstrated in Figure 2.1 using the Lena image.

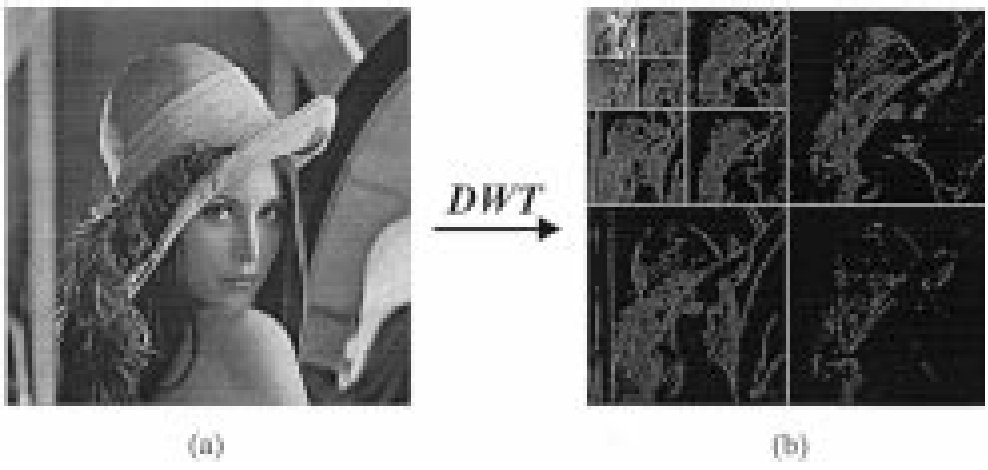

Figure 2.4 (a) The original 512x512 Lena image. (b) DWT decomposition of the Lena image [12].

One way of inserting a watermark at this stage is by using wavelet-based fusion where wavelet coefficients of the watermark and the image are added at different resolution levels. Before the watermark wavelet coefficients are added, they are modulated using a human visual-model constraint [10]. Once the casting positions are chosen and the watermark is embedded into the different subbands, the inverse discrete wavelet transform (IDWT) is performed to obtain the watermarked image. This embedding process is depicted in Figure 2.5.

Watermarking in the wavelet domain provides robustness to JPEG compression and many schemes that employ DWT for watermarking have been developed.

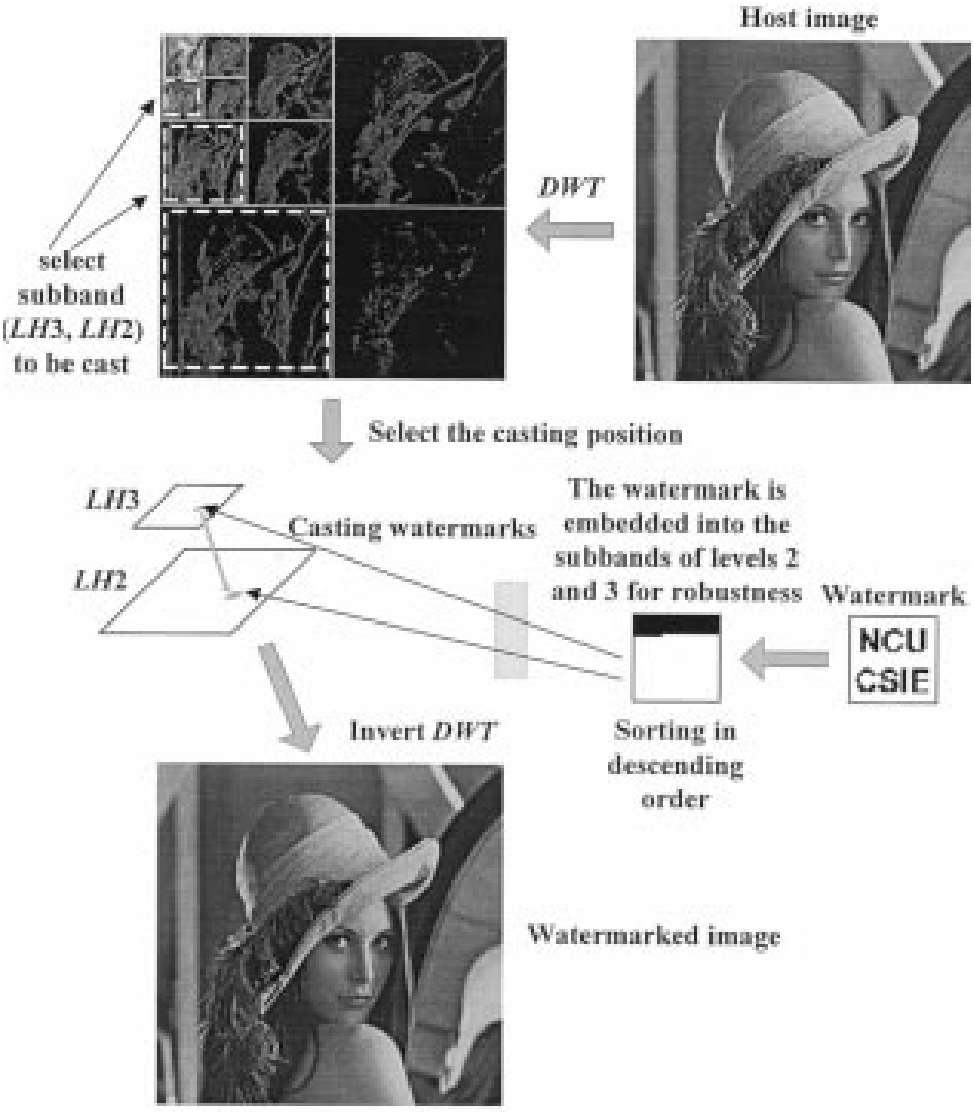

Figure 2.5 General DWT watermark embedding process [12].

2.3.2.3. Discrete Fourier Transform

The Discrete Fourier transform is widely studied in the signal processing field and was considered in the field of watermarking in order to present the possibility of controlling the frequencies of the host signal. It is often useful to select the adequate parts of the image for embedding the watermark in order to obtain the best compromise between visibility and robustness [10].

The two-dimensional DFT is performed in the real number arena and thus the DFT of an image results in the magnitude and phase representation of the image. It was thus found that the phase modulation could be used for robust watermarking [10]. The phase components of the DFT have a more psychovisual impact than the magnitude components. Thus when the watermark is embedded in the phase components with high redundancy, a malicious attacker would cause unacceptable damage to the quality of the image in trying to remove the watermark. This aspect of the DFT coefficient guarantees that attacks against the watermark will cause unacceptable loss of the image quality.

DFT proves to be useful for watermarking purposes where phase modulation is performed between the watermark and its cover and the DFT is also used to split images into perceptual bands.

2.3.3. Watermark attacks

As with any security measure or system, watermarks are subject to certain attacks. These attacks can be categorized into four main classes namely removal attacks, geometric attacks, cryptographic attacks and protocol attacks. It is important to bear in mind when exploring the different attacks that robustness requirements for each watermarking system are application dependant and may not cater against all the possible attacks.

2.3.3.1. Removal Attacks

The removal attack, also known as the robustness attack, aims to remove the presence of the watermark without harming the image beyond rendering it useless. This attack involves removing the watermark information from the watermarked data without cracking the security of the watermarking algorithm, i.e. without the key used for watermark embedding [13]. This implies that no processing, however complex, should be able to recover the watermark information from the attacked data.

Typical removal attacks include denoising, quantization (for lossy compression), averaging, filtering, printing and scanning [13]. Although all these methods aim to achieve complete watermark removal, this is not always the case but most methods do significantly damage the watermark information. In more complex removal attacks, statistical models for the original image and the watermark are exploited in an effort to optimize operations such as denoising or quantization so as to destroy the embedded watermark while maintaining the quality of the image.

Generally, the effect of removal attacks results in a decrease of the effective channel capacity in the process of removing the embedded watermark.

2.3.3.2. Geometric Attacks

Geometric attacks, also known as presentation attacks, manipulate the image so that the detector can't find the watermark anymore. Examples of such an attack are scaling, cropping, rotation and jittering. Unlike removal attacks, geometric attacks do not actually remove the watermark but instead distort the watermark so that detection is impaired. Thus the watermark detector synchronization is lost with the embedded information [13].

In image watermarking, the best known benchmarking tools are Stirmark and Unzign. Both these benchmarks make use of a variety of geometric attacks. Unzign introduces local pixel jittering and has proved to be efficient in attacking spatial domain watermarking methods. Stirmark tends to introduce global and local geometric distortions. Most modern watermarking techniques survive these attacks by using special synchronization techniques. Although resistance to the random bending attack presented by Stirmark still remains a problem for most commercial watermarking tools [13]. This particular random attack exploits the aspect that the human visual system (HVS) is insensitive to local shifts such as pixels that are shifted, scaled and rotated without introducing considerable visual distortion.

Geometric attacks can sometimes also be referred to as detection-disabling or synchronization

attacks, since the main aim is an attempt to break the correlation and make the recovery of the watermark infeasible for the watermark detector. Thus in this type of attack the watermark still remains in the attacked data but cannot be detected. An exception to this is if the watermark detector or decoder is increased in intelligence or complexity.

2.3.3.3. Cryptographic Attacks

Cryptographic attacks intend to crack the security methods in watermarking schemes and thus obtain a way to remove the embedded watermark or to embed misleading watermarks. An example of a cryptographic attack is the brute-force key search for the embedded secret information. Another such attack is the Oracle attack which can create a non-watermarked signal when a watermark detector device is at hand.

The goal of cryptographic attacks is to remove or destroy the embedded watermark but the application of these attacks is restricted due to their high computational complexity.

2.3.3.4. Protocol Attacks

Protocol attacks aims at attacking the entire concept of the watermarking application by identifying weaknesses on a system level and then showing that the given watermarking method is not secure.

One type of protocol attack is the copy attack where the aim is not to destroy the watermark or impair its detection but rather to estimate the watermark from the watermarked data and copy it to some other data, usually called target data [13]. The estimated watermark is adapted to the local features of the target data in order to satisfy its imperceptibility. Copy attacks are applicable when a valid watermark in the target data can be produced with no knowledge of the watermarking algorithm or the key used to embed the watermark.

Another protocol attack is known as the inversion attack where the attacker subtracts his/her

own watermark from the watermarked data and claims to be the owner of the watermarked data. This causes ambiguity as to who the true owner of the image is. It is often required, for copyright protection applications, that watermarks should be noninvertible. This implies that it should not be possible to extract a watermark from a non-watermarked document.

2.3.3.5. Other Attacks

Other attacks that are commonly found are estimation-based attacks and interpretation attacks. Estimation-based attacks are based on the knowledge of the watermarking technology and exploit statistics of the original image and the watermarked data. Interpretation attacks are an attempt to introduce another watermark in an already watermarked image, thereby creating an ownership deadlock.

It can thus be seen, that it is important to take attacks into account during the design process of a watermarking system.

2.4. WATERMARKING METHODS FOR JPEG IMAGES

This section provides a brief description of a couple of the current watermarking methods used for the protection of JPEG images. The focus is on JPEG images since this dissertation aims develop a watermarking algorithm for JPEG images.

In JPEG-to JPEG image watermarking (J2J), the input file is a JPEG image file and the watermarked image is JPEG compressed such that the output file is also a JPEG file. One such technique uses the human visual system (HVS) to estimate the maximum number of bits that can be embedded in JPEG compressed images. In this particular method Watson's HVS model is modified to estimate the just noticeable difference (JND) for DCT coefficients [14]. The amount of modifications that can be performed on DCT coefficients is limited by the JND in order to guarantee the invisibility of the watermark.

There are a number of existing J2J watermarking schemes. Some schemes use the inter-block correlation of selected DCT coefficients to embed the watermark bits [15, 16]. This is achieved by adding or subtracting an offset to the mean value of the neighboring DCT coefficients. Other proposed schemes hide the watermark bits by modifying the DC [17] and AC [18] coefficients in the block-based DCT domain. There are also methods that protect JPEG images by using the modified discrete cosine transform (MDCT). These schemes were shown to achieve a high image quality as well as robustness to compression, filtering and geometric transformations.

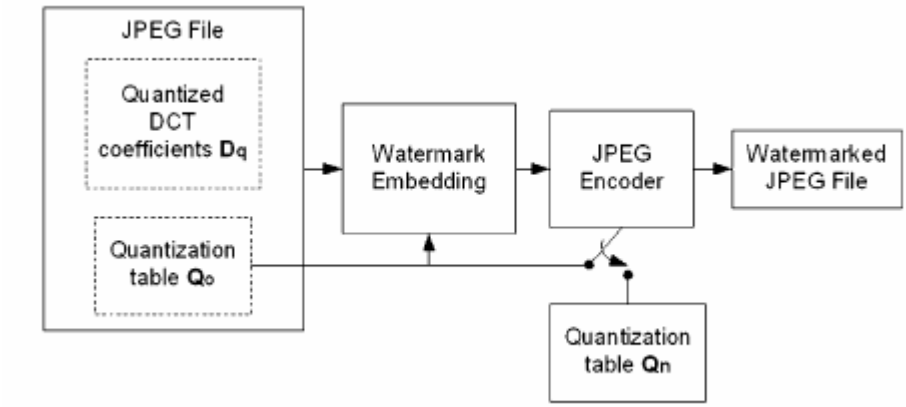

Figure 2.6 JPEG-JPEG watermarking (J2J) [14]

The JPEG file needs to be partially or fully decoded in order to embed a watermark in JPEG-compressed images. The level of decoding is dependant on the domain in which the watermark will be embedded in. Should the watermark be embedded in the bitstream domain, only variable-length decoding is required. If the watermark is embedded in the 8x8 block-based DCT domain, then inverse zigzag scanning and inverse quantization will be required. In the case where the watermark is embedded in the spatial domain or any frequency domain, inverse DCT would be necessary.

A generalized J2J watermarking model is depicted in Figure 2.6. In this figure, it is assumed that the watermarked images will be JPEG-compressed using either the original quantization table or a new quantization table defined by the user. It is also assumed that the dimensions of

the images are not changed in the watermark embedding process.

From the above discussions and current literature, it can be seen that most JPEG watermarking schemes tend to mostly make use of the transform or spread spectrum domain to embed a watermark. It is also noted that most schemes aim to keep the image dimensions the same while introducing some percentage of distortion. Thus there is an opening in this field for a new watermarking scheme which embeds a watermark using encoding techniques instead of spectral techniques. This forms goal of this dissertation and it is also aimed to achieve a watermarked image which is perhaps slightly expanded but does not introduce any distortion.

2.5. SUMMARY

Digital watermarking is a means for the copyright protection of digital images. There are several aspects of a watermarking scheme which are explored in this chapter. An overview of the manner in which a watermarking system functions and its requirements is given. Aspects such as the main watermarking domains and attacks that a watermark is subject to are also discussed.

This chapter thus provides a comprehensive study of the current watermarking techniques, the attacks on watermarking schemes and an overview of the most prominent JPEG watermarking techniques.

CHAPTER 3. JPEG COMPRESSION STANDARD

3.1. JPEG OVERVIEW

As digital technology has advanced at such a rapid pace, storage and communication of digital material, in particular digital images poses a problem. In order to address this, compression is used. There are two main categories of compression, namely lossy and lossless compression.

Lossless compression algorithms e.g. GIF, BMP and RLE reduce the size of the file by relying on elimination of the redundancy in the file. It is also known as the noiseless coding, entropy coding, and data compaction codes and these algorithms can perfectly recover original image.

Lossy compression implies that the reconstructed image will look almost identical to the original image but is not identical and allows for a significant reduction in size. Most images contain a lot of redundant information to the human eye, which is sensitive to some types of distortion and less sensitive other types of distortion in an image. Good compression algorithms comprehend how human perception works [7] and this should be taken into consideration in the design of a lossy image compression algorithm.

JPEG presents a simple lossy technique referred to as the Baseline method. Algorithms such as JPEG discard much of the original information from an image such that you cannot reconstruct the exact original image from a JPEG file [66]. This is information that is not really noticed by human users and can thus be discarded for compression purposes. JPEG is mainly used on continuous tone images such as photographs. Since this kind of images do not contain much redundancy, pure lossless compression does not achieve significant reductions in size [22].

The JPEG standard has become a popular format for images on the internet and digital images produced by digital cameras are often stored in the JPEG format. JPEG can obtain higher compression ratios on images with high pixel resolutions, simply because the degradation isn't as noticeable on a big image. In the case of JPEG, the loss of specific picture detail is not detectable to the untrained human eye until compression ratios reach a high of 80-90 percentage ranges [68]. The noticeable distortion is normally called an artifact. The following section details the aspects of JPEG compression.

3.2. THE JPEG IMAGE-CODING STANDARD

The JPEG standard, adopted by the Joint Photographic Experts Group, is a widely used standard for lossy compression of still images [11]. The JPEG standard can be applied to both grayscale and colour images and achieves very good to excellent image quality. This standard can be implemented in software with an acceptable computational complexity.

JPEG first converts the image to be compressed into the YCbCr colour space and each colour plane is broken into 8x8 pixel blocks [20]. The transformation into the YCbCr will not be explored as this dissertation will focus on single component (grayscale) image compression.

The blocks are then DCT transformed and the DCT coefficients are divided by some predefined quantization values and rounded to the nearest integer (according to a quality factor, the quantization values can be scaled by a constant) [10]. The resulting quantized DCT coefficients are compressed by means of an entropy encoder such as Huffman or arithmetic coding. An inverse DCT is then performed to reconstruct the data resulting in an image very similar to the original one. If the quantization values were set properly, the human eye should perceive no visible difference.

The JPEG encoder consists of three main blocks as depicted in Figure 3.1.

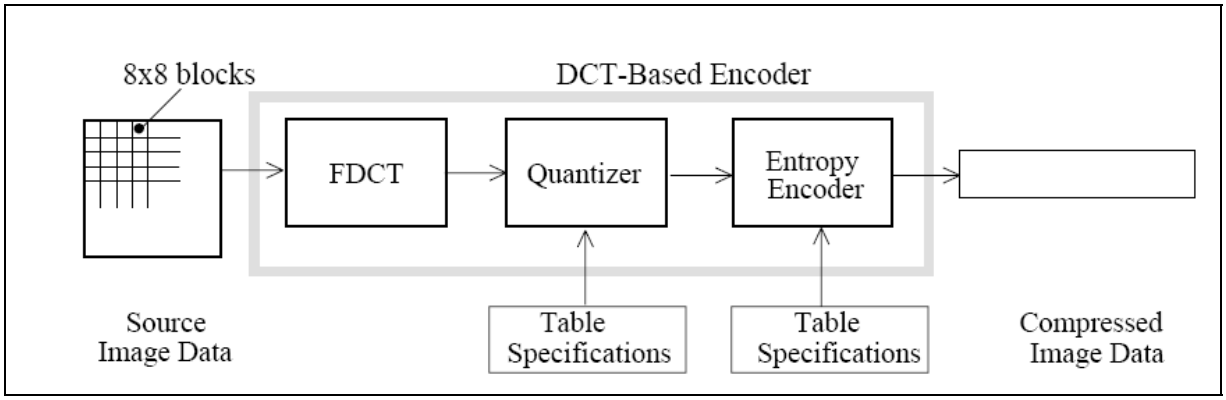

Figure 3.1 JPEG: DCT-Based Encoder Processing Steps [19]

The Decoder process is the inverse of the encoder process and is depicted in Figure 3.2 below. This uses the compressed image data to reconstruct the image using the inverse processes such as the Inverse Discrete Cosine Transform (IDCT).

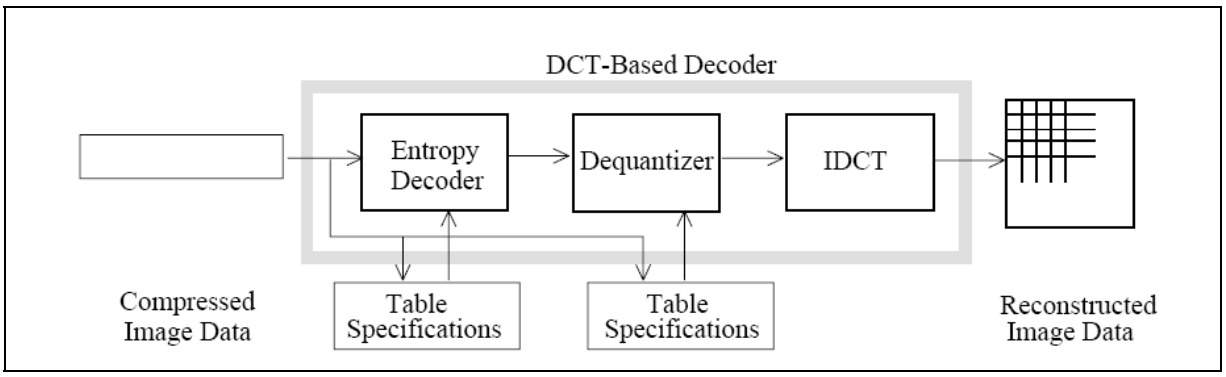

Figure 3.2 JPEG: DCT-Based Decoder Processing Steps [19]

This gives an overview of the processing involved in the JPEG standard. The following sections explore each processing step in detail and how each step contributes to the final outcome of the JPEG image-coding standard [24].

3.3. THE DCT COMPONENT

JPEG makes use of transform coding techniques. Transform coding techniques compress the

transform of a signal i.e. an image, and not the image directly. The Discrete Cosine Transform (DCT) is the most commonly used transform technique for image coding [11]. The energy compaction property of the DCT results in transform coefficients with only a few of the coefficients having significant values, thus making it a popular technique.

Each colour component of a continuous tone image can be represented as a series of amplitudes in the two dimensional space. The DCT is used to discard higher frequency information that has minimal visual effect on the image. The DCT is related to the Discrete Fourier Transform (DFT) but can approximate linear signals well with few coefficients.

An image consists of many pixels arranged in an m x n array. The first step in the Forward Discrete Cosine Transform (FDCT) transformation of the image is to divide the picture array into 8x8 blocks of pixels. The size of the blocks has been chosen as a compromise of complexity and quality. If the number of rows or columns is not a multiple of 8, the closest multiple of 8 rows or columns is considered when dividing the image into 8x8 blocks. A comparison of the colour data but not the intensity level of each pixel to its neighbors is performed. The DCT produces a large discrete coefficient for a pixel if the differences between the pixels is large else if the difference is little, a small DC coefficient. The large amplitudes signify a large colour difference. The pixels have thus been mapped from the spatial domain to the frequency domain.

This is in essence where the fundamental principle of JPEG lies, in eliminating the subtle colour changes that the human eye cannot detect. In the frequency domain, this maps to the smaller coefficients and thus these coefficients can be eliminated or rounded to zero. The data which contains the significant information is then retained by keeping the medium and larger coefficients.

In order to calculate the two-dimensional DCT coefficients, S(u, v), the general equation is given as

$$S(u, v) = \frac{2}{N} C(u)C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} s(x, y) \cos\left(\frac{\pi u(2x+1)}{2N}\right) \cos\left(\frac{\pi v(2y+1)}{2N}\right), \quad (3.1)$$

where N is the number of blocks of pixels according to which the image will be divided. C(u) and C(v) take the value of $1/\sqrt{2}$ when u = 0 and v = 0 respectively [10]. C(u) and C(v) take the value of 1 otherwise. s(x,y) represents the values of the 8x8 subarray block of whose DCT is being calculated. The mapping from the spatial domain to the frequency domain as a result of the DCT is shown in Figure 3.3.

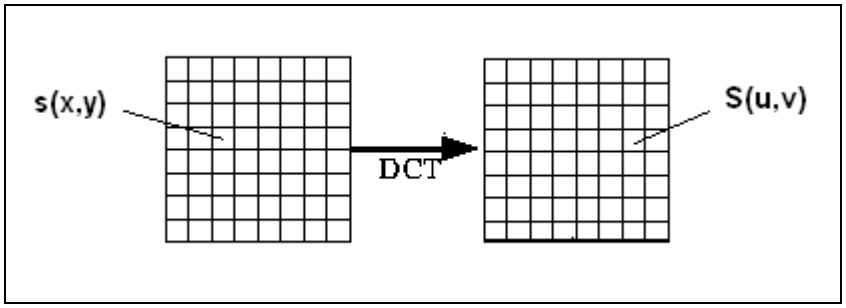

Figure 3.3 Spatial domain to Frequency domain mapping

In computing the DCT of each subarray, 64 DCT coefficients are generated. S(0,0) is known as the DC component and the remaining components are referred to as AC components.

At the decoder the IDCT reverses this processing step by reconstructing a 64 point output image signal by summing the basis signal [19]. In order to perform the reverse mapping from the frequency domain to the spatial domain, the following equation is used:

$$s(x, y) = \frac{2}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} C(u)C(v)\, S(u, v) \cos\left(\frac{\pi u(2x+1)}{2N}\right) \cos\left(\frac{\pi v(2y+1)}{2N}\right) \quad (3.2)$$

This DCT processing step is then followed by the quantization process which is explored

below.

3.4. THE QUANTIZER

The completion of a DCT results in the low frequency components of the coefficients having significant values. The DC component contains a significant amount of energy and there is a correlation between DC components of the current and preceding subarrays. A Uniform differential quantization scheme is used for the quantization of DC components and AC components are quantized using uniform quantization schemes. This is detailed later in this section.

The purpose of quantization is to modulate the influence of the different spectral components on the image. The influence of the highest DCT coefficients, which normally contain noise, is reduced. In essence each block is quantized using a quantization table meaning that each DCT coefficient is divided by its corresponding quantizer step size (an integer between 1 and 255) and rounded to the nearest integer. This is mathematically computed using the following equation.

$$S_q(u,v) = \text{Round}_{\text{integer}} \left(\frac{S(u,v)}{Q(u,v)} \right), \tag{3.3}$$

where Sq(u,v) is the resulting quantized coefficient, $S(u,v)$ is the DCT coefficient and $Q(u,v)$ is the quantizer step size as per the quantization table.

JPEG compresses the color information, or "chrominance", in an image separately from the actual details of shapes, or "luminance". Luminance amounts to a grayscale image, while the chrominance amounts to a wash of colors painted on top of that grayscale image. The eye is much more sensitive to the details of shapes than color information, so the chrominance information can be compressed to a greater level than the luminance information. This means that JPEG will usually have higher compression ratios for color images than for grayscale

images. An example of a luminance quantization table is depicted below as this dissertation addresses grayscale images in particular.

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

Figure 3.4 Quantization Table [21]

The quantization step discards the information which is not visually significant and is thus the key source of lossiness in DCT-based encoders. The selection of the quantization table presents a trade-off between the compression achieved and the quality of the image. Dequantization is the inverse function which involves multiplying by the quantizer step size as indicated in equation below. This results in a form appropriate as input to the IDCT. The quantization table is stored in the file header for use by the decompressor.

$$S_q'(u,v) = S_q(u,v) * Q(u,v) \tag{3.4}$$

Quantization thus enables lossy compression of an image by representing DCT coefficients with no greater precision than is essential to achieve the required image quality.

Once the quantization process has been performed, the DC coefficient is encoded as the difference from the DC term of the previous bock in the encoding order as depicted in Figure 3.5 since there is a strong correlation between the DC coefficients of adjacent 8x8 blocks.

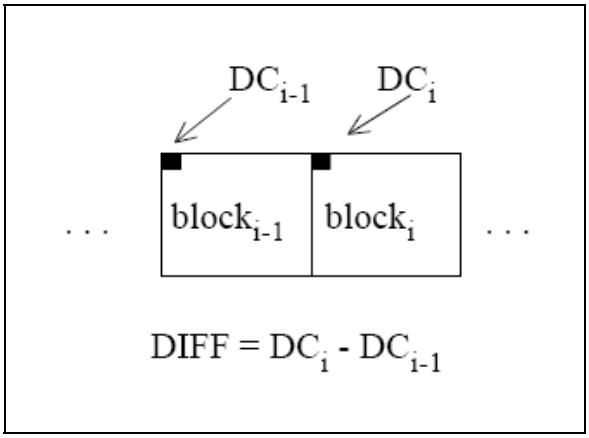

Figure 3.5 Differential DC Encoding [19]

The quantized DCT coefficients are arranged in a vector by zig-zag sampling as shown in the following figure. This sampling results in a vector of length 64 with non-zero values populating only the first few components. This enables to facilitate entropy encoding by placing low frequency coefficients before high frequency coefficients.

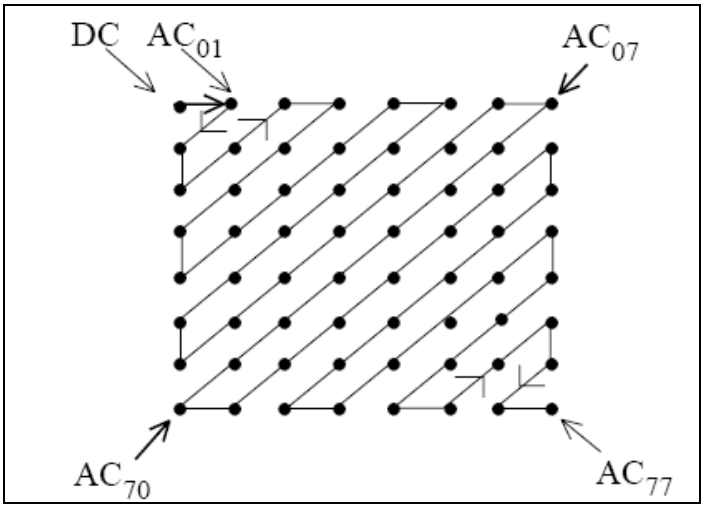

Figure 3.6 Zig-Zag Sampling [19]

3.5. THE ENTROPY ENCODER

Entropy, in the context of information theory refers to information content or 'density'. To grasp the concept of entropy consider the following explanation.

Consider a source S = {b_1, b_2, ……, b_n}. Each element b_i is a random variable or a symbol. Assume successive symbols are independent, and the probability of the occurrence of the symbol b_i is given by $P(b_i)$. If symbol b_i occurs, an amount of information equal to:

$$I(b_i) = \log_2 \left(\frac{1}{P(b_i)} \right) \text{ bits} \tag{3.5}$$

is provided. The entropy or the average amount of information per source symbol is:

$$H_1(S) = -\sum_{i=1}^{n} P(b_i) \log_2 P(b_i) \tag{3.6}$$

This is called the first order entropy of the source S.

Entropy represents the amount of information associated with a source (image) or a degree of randomness (uncertainty) in a signal. It also represents the smallest number of bits needed on average to represent symbol average on all symbols code lengths.

The process of entropy coding can be split in two parts: modeling and coding. Modeling assigns probabilities to the symbols, and coding produces a bit sequence from these probabilities. As established in Shannon's source coding theorem, there is a relationship between the symbol probability and its corresponding bit sequence. A symbol with probability p gets a bit sequence of length -log(p). In order to achieve a good compression rate, exact probability estimation is needed. Since the model is responsible for the probability of each symbol, modeling is one of the most important tasks in data compression.

Entropy encoding allows for the lossless compression of the quantized values. This is achieved by encoding the quantized DCT coefficients more compactly based on their statistical characteristics.

The JPEG standard allows for two options namely Huffman coding and Arithmetic coding. The Baseline sequential mode makes use of Huffman coding, the extended sequential mode can make use of Arithmetic coding.

Entropy Coding is a form of compression that relies on developing "code" words in a library for the input values. It consists of two steps which are explored here. The first step entails converting the zig-zag sequence of quantized coefficients into an intermediate sequence of symbols. The second step then involves converting these symbols to a data stream in which the symbols no longer have externally identifiable boundaries. The form of the intermediate symbols is dependant on both the DCT-based mode of operation and the entropy encoding method [19].

Huffman coding usually requires that sets of Huffman code tables to be specified by the application and these tables are utilized in the compression and decompression of an image [23]. These tables may be pre-defined or computed specifically for the image at hand in an initial statistics-gathering pass prior to compression.

Arithmetic coding on the other hand requires no external tables to be input as it adapts to the image statistics as it encodes the image. Arithmetic coding is the entropy encoding scheme used in this implementation and an expansive explanation of how Arithmetic coding works is given in the following Chapter of this dissertation.

Although Huffman is simpler, Arithmetic coding performs better than Huffman coding with regards to speed and efficiency. Arithmetic coding has achieved 5-10% better compression than Huffman for several images tested [19]. Arithmetic coding gains an advantage by being ‘adaptive’ i.e. being able to produce probability estimates on the go where as Huffman coding

requires a set of constant probabilities for each of the symbols prior to encoding.

3.6. PERFORMANCE CRITERIA FOR IMAGE COMPRESSION

The goal of image compression is to transform an image into compressed form so that the information content is preserved as much as possible. It is fairly simple to design a compression algorithm that achieves a low bit rate, but the challenge is how to preserve the quality of the reconstructed image at the same time. The two main criteria of measuring the performance of an image compression algorithm are compression efficiency and distortion caused by the compression algorithm. Other criteria that play a role are the speed and complexity of the compression algorithm. That refers to the speed of the compression and decompression process and the ease of implementation of the algorithm. Often the reliability of software depends on the complexity of the algorithm.

The way in which some of these performance criteria will be measured is investigated and explored in the Chapter 5 of this dissertation.

3.7. JPEG COMPRESSION AND IMAGE QUALITY

JPEG is capable of achieving high compression ratios with good image quality for both grayscale and colour images. This is dependant on the rates, measured in bits/pixel, used for each of the varying applications. This unit of measure implies the total number of bits in the compressed image. The following levels are a guideline of the picture quality for the indicated ranges of compression when applying DCT-based modes of operation [11],[19]:

- 0.25 – 0.5 bits/pixel: moderate to good quality pictures can be obtained that are sufficient for some applications;
- 0.5 – 0.75 bits/pixel: good to very good quality images that are sufficient for many applications;
- 0.75 – 1.5 bits/pixel: excellent quality images are obtained sufficient for most applications;

- 1.5 – 2.0 bits/pixel: practically indistinguishable from the original and sufficient for the most demanding applications.

Most software that can create JPEG images, such as paint programs, allows the user to set a "quality factor" to specify the percentage level of degradation, thus compression ratios can be increased if a higher degradation is tolerable. This quality factor is a more or less random scale and may not give exactly the same results in different software packages, but as a rule a 75% percent quality factor gives a perfectly acceptable image, while quality factors above 95% buy very little improvement in the image, and quality factors below 10% produce a completely unacceptable image.

3.8. SUMMARY

It has been seen that there is a need for the copyright protection of digital images [69] particularly for images on the Internet. This chapter encapsulates the details of the processing steps involved in the JPEG compression standard and the constituents to measure the compression performance and image quality. The mechanism of each processing step in the compression algorithm is explored and this gives a clear indication of how the JPEG compression algorithm works.

The JPEG image compression standard provides an efficient means of reducing the size of continuous-tone images. This serves as an immense benefit for the storage and communication of digital images and thus implies a cost benefit as well.

CHAPTER 4. ARITHMETIC AND HOMOPHONIC CODING OVERVIEW

The underlying idea of the proposed algorithm is to combine entropy coding (Arithmetic coding) with homophonic coding while embedding a watermark in the process such that no distortion is introduced into the image once the watermark is inserted. In combining the two coding methods, it is essential to understand the workings of each and how they possibly interrelate.

4.1. ARITHMETIC CODING

Arithmetic coding [27][25] is one of the entropy encoding methods that can be utilized by the JPEG image coding standard. Entropy coders achieve compression by exploiting the nonuniformity in the probabilities under which a signal to be compressed takes on its allowed values [30]. Arithmetic coding [25] [27] is derived from the Elias coding [31] in which the coded message can be taken as a single long binary fraction.

Arithmetic coding encodes data by creating a code string which is represented by a binary fraction on an interval between 0 and 1. The coding algorithm encodes/decodes one data symbol per iteration [26]. A simple arithmetic coding algorithm is investigated in more detail in Section 4.1.3. It should be taken into consideration that there are several arithmetic coding algorithms which can be used. In this dissertation we focus on a simple arithmetic coding algorithm while touching on others such as the shift and add method [34].

The data models used by Arithmetic coding can be one of the following:

- Adaptive which dynamically estimating the probability of each event based on all

events that precede it;

- Semi-Adaptive by using a preliminary pass of the input file to gather statistics;
- Non-Adaptive which uses fixed probabilities for all files.

Arithmetic coding is efficient in both theory and practice as the data model used with arithmetic coding can be adaptive. This is particularly useful for image sources and this gives good compression. Adaptive arithmetic coding is a one pass process in which the coder builds up the statistical model while coding the input data. Static (semi-adaptive) entropy coders need two passes or make use of statistical models.

Binary Arithmetic Coding (BAC) [61] is the most effective statistical coding method used in image compression. Despite the wide use of this entropy encoder, arithmetic coders are generally incompatible with each other thus application developers are faced with the difficult task of creating each coder.

Arithmetic coding uses an interval to represent the probability of a sequence i.e. it uses a number in the interval to differentiate the sequence from all other possible sequences. Arithmetic coding is usually implemented with incremental transmission and reception with arithmetic operations of fixed precision [32]. Fixed precision arithmetic coding simplifies software and hardware implementation and reduces complexity at the same time.

4.1.1. Advantages

Although Arithmetic coding is more complex than Huffman coding, it offers some important advantages which are highlighted below:

- Coding efficiency is higher than other entropy coding methods.

- As a source coding scheme, arithmetic coding is able to approach the source's entropy arbitrarily close [26][27].

- Coding and source modelling are separate i.e a clear separation exists between the modelling of source statistics and the encoding process of the source symbols. This implies that an arithmetic coder can work in collaboration with any model that can provide a sequence of event probabilities. This can be of great benefit as large compression gains can be achieved through the use of complex models that provide more accurate probabilistic descriptions of the input data [28][32].

- The algorithm is easily adaptable to varying source statistics [34].

- Unlike Huffman coding, it is not required to arrange the symbol probabilities in any particular order.

The use of an arithmetic coder reduces the overall resulting size of the JPEG data by a further 10 percent to 15 percent over the results that would be achieved by the Huffman coder. With no change in resulting image quality, this gain could be of importance in implementations where enormous quantities of JPEG images are archived.

4.1.2. Disadvantages

Arithmetic encoding has a few disadvantages that are listed here:

- Not all JPEG decoders support arithmetic decoding. Baseline JPEG decoders are required to support only the Huffman algorithm.
- The arithmetic algorithm is slower in both encoding and decoding than Huffman.
- Arithmetic coding is slightly more complex to implement in comparison to Huffman coding implementation.
- The arithmetic coder used by most JPEG algorithms (called a Q-coder [39] - [41]) is owned by IBM and AT&T (Mitsubishi also holds patents on arithmetic coding). It is required to obtain a license from the appropriate vendors if their Q-coders are used for the back end of a particular JPEG implementation.

4.1.3. Arithmetic coding Algorithm

Arithmetic coding maps a string of source symbols to a code string such that the original data can be recovered from the code string [26]. Arithmetic operations are performed on the code string in both the encoding and decoding algorithms.

Any coding technique makes use of a data model [43]. The model determines the following:

- the symbols to be encoded; and
- the estimate of the probability (relative frequency) of the symbols.

The estimation method computes the relative frequency distribution for each context [26]. This can be computed before hand (fixed) or may be performed during the encoding process (adaptive). The memory-less model is an example of a simple model where the data symbols themselves are encoded to a single code [26]. A different model such as the first-order Markov model [30] uses the previous symbol as the context for the current symbol. The context provides a probability distribution for the encoding of the next symbol and thus makes the Markov model a dependent model as the context is governed by the past sequence of symbols.

The encoder then receives the symbol and its probability to generate the code string. Arithmetic encoding involves the repeated division of an interval into sub intervals with widths corresponding to the probabilities of the input symbols. The codewords representing the source symbols can be seen as code points on a unit interval. The codewords divide the unit interval into non-overlapping subintervals. Probabilities are expressed as binary fractions and each codeword is equal to the cumulative sum of the probabilities of the preceding symbols. The encoded data is taken as the lower bound of the final interval.

For the purposes of this dissertation, the adaptive single-context model or memory-less model is considered. A conceptual description of exactly how this Arithmetic coding algorithm is used to encode a file is explored here.

The probability table assigns a range between 0 and 1 to each input character. The size of each range is directly proportional to a characters' frequency. The order of assigning these ranges is not as important as the fact that it must be used by both the encoder and decoder. The range consists of a low value and a high value. These parameters are very important to the encode/decode process. The more frequently occurring characters are assigned wider ranges in the interval requiring fewer bits to represent them. The less likely characters are assigned more narrow ranges, requiring more bits.
The basic algorithm for arithmetic coding involves the following [29]:

- It begins with a "current interval" [L, H) initialized to [0, 1).
- For each symbol in the file, two steps are performed.
    - The current interval is subdivided into subintervals, one for each possible symbol. The size of the symbol's subinterval is proportional to the estimated probability that the symbol will be the next symbol in the file, according to the model of input.
    - The subinterval corresponding to the symbol that actually occurs next is selected and is made the new current interval.
- The output has enough bits to distinguish the final current interval from all the other possible final intervals.

The length of the final subinterval is clearly equal to the product of the probabilities of the individual symbols, which is the probability p of the particular sequence of symbols in the file. The final step uses at the most $[-\log_2 p] + 2$ bits to distinguish the file from all other possible files. A method of indicating the end of the file is needed, usually by a special end-of-file symbol coded just once or some external indication of the file's length. This usually only adds a small amount to the code length. In the second step, it is required to compute only the subinterval corresponding to the symbol a_i that actually occurs. This is usually achieved by using two cumulative probabilities. The cumulative probability as given by the equation below:

$$P_C = \sum_{k=1}^{i-1} p_k \tag{4.1}$$

The next cumulative probability is given respectively by the equation below:

$$P_N = P_C + p_i = \sum_{k=1}^{i} p_k \tag{4.2}$$

The new subinterval is obtained by the following equation:

$$[L + P_C(H-L), L + P_N(H-L)) \tag{4.3}$$

The need to maintain and supply cumulative probabilities requires the model to have a complicated structure especially when more than two events are possible.

The encoding process amounts to the adding of properly scaled cumulative probabilities P_C, also known as augends, to the code string.

In the instance where the cumulative probabilities of the source symbols are dyadic i.e. negative powers of two [4], arithmetic coding can be simplified to the sum of the augends. This implies that the coding can be performed using shifting and adding operations instead of dividing and re-scaling operations. When using the shift and add algorithm for arithmetic coding [4], sometimes a problem known as the carry-over problem [55] is encountered. This is when all the bits in the codeword are changed due to the carry-bit when adding and has a ripple effect on the codeword bits that have already been computed. This problem is easily resolved by means of bit-stuffing [25], [57].

4.1.3.1. Arithmetic Coding Example

An example of pure Arithmetic coding is shown below [32]:

- Let the message be *ABRA/CADABRA*, and suppose that the symbol probabilities are as follows:

Table 4.1 Symbol probabilities for Arithmetic example

/	0.1
A	0.4
B	0.2
C	0.1
D	0.1
R	0.1

- The probabilities will always sum to 1 across the complete alphabet, and each symbol is allocated a section of the scale from 0-1 according to its probability, as shown in the figure.

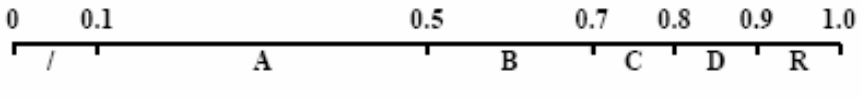

Figure 4.1 The Arithmetic code probability scale

- The codeword is a single number within the range 0-1, and is constructed as shown in the following figure:
- The first symbol (A) defines the range from 0.1-0.5 on the initial scale, and the codeword is within this range.

- The second symbol (B) then subdivides that range according to its range of 0.5-0.7, so the codeword now lies between (0.1 + 0.4 x 0.5) and (0.1 + 0.4 x 0.7), i.e. 0.3-0.38.
- The next symbol subdivides the codeword range according to its probability, & etc.
- Thus after the sequence **ABRA/CA** the codeword would be within the range 0.3730272-0.3730400. If this were the end of the message, the actual codeword would be taken as the lowest precision binary number lying within this range – **0.01011111011111112.**

  (This could of course be scaled by 2^{16} and transmitted as 0101111101111111).
- For comparison, an equal length coding of the 6-symbol alphabet would require 7x3bits = 21 bits for this message, so already it can be seen that a saving of 5/21 or 24% is achieved.

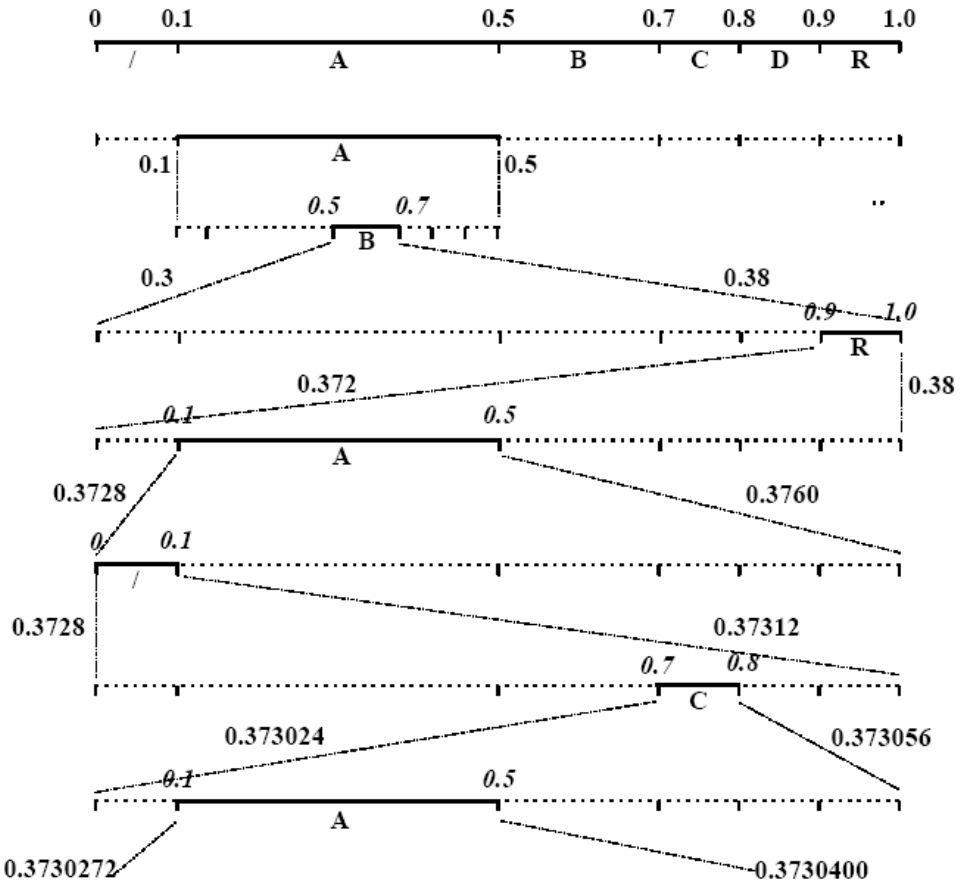

Figure 4.2 The Arithmetic coding process for the sequence ABRA/CA

Decoding simply requires the algorithm to be applied in reverse: the value of the received codeword defines the range of the first symbol, the codeword is then rescaled within the probability range of the first symbol to identify the second symbol, etc.

Further arithmetic encoding examples can be found in [4] and [26].

The decoder operation of arithmetic coding recursively reverses the encoder operation by the following steps [4]:

- Decoder comparison;
- Decoder readjust; and
- Decoder scaling.

4.2. HOMOPHONIC CODING

Homophonic coding is known to be a kind of message randomisation used to make a code sequence indistinguishable from a sequence of completely random digits. Such sequences are of great interest for cryptography since they permit constructing unbreakable cipher systems.

Over the years that cryptology has been present, it can be seen that the majority of secret-key cipher systems were broken by making use of the fact that plaintext characters are not uniformly distributed. In simple substitution each plaintext source symbol is replaced with a corresponding codeword by using one-to-one mapping [34]. Homophonic coding or homophonic substitution is a method used to convert a given plaintext sequence into a random sequence thus thwarting cipher-text only attacks and adds to the reliability of secret-key cipher systems [36].

Homophonic coding makes use of one-to-many mapping where each source symbol is mapped into a set of codewords referred to as homophones. Source symbols which occur frequently are mapped into more than one code word, in order to flatten the frequency distribution of the resulting codewords [36]. If the number of homophones (symbols) assigned to each source

input is proportional to its probability, then single-source frequency information is completely obliterated [35].

The main aim is to convert a source message, consisting of non-uniformly distributed letters of some alphabet, into a uniquely decodable sequence of uniformly distributed code symbols. The uniformity of distribution means that all the symbols are equiprobable and independent, i.e. completely random.

Homophonic coding consists of a message source, a reversible memoryless mapping and a source encoder. The required mapping is performed by the homophonic channel whose output is then compressed by a suitable source encoder. The homophonic channel and the source encoder is jointly referred to as the homophonic coder [34]. Under normal circumstances, an external randomizer provides the required random bits for the selection of the homophones.

The schematic diagram of a general homophonic coding scheme is shown in Figure 4.3.

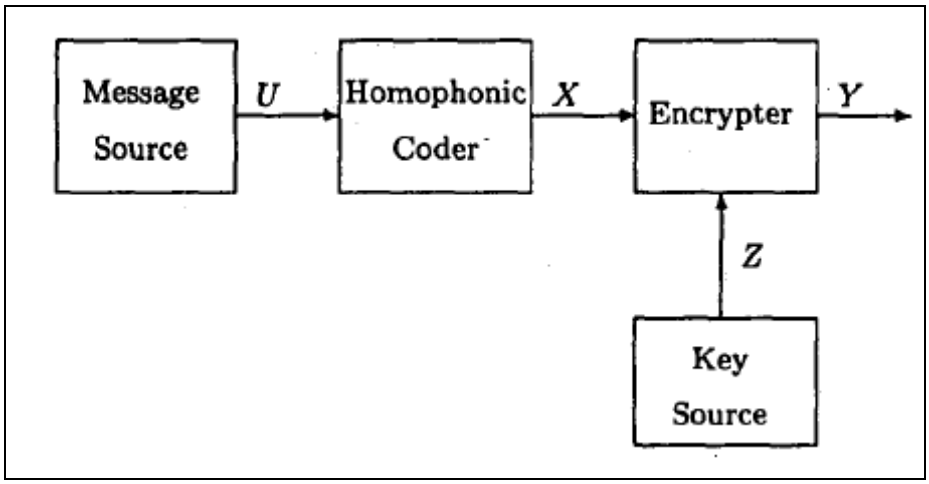

Figure 4.3 Block Diagram of a general homophonic coding scheme [34]

In the above diagram [34], the message source U produces a sequence of random variables $U = \{u_1, u_2, \ldots, u_L\}$, where $2 \leq L \leq \infty$, as per the probability distribution $P(u_1), P(u_2), \ldots, P(u_L)$. The

homophonic coder maps the output of the message source into the sequence $X_1,X_2,\ldots X_n$. In this case this will be regarded as the plaintext sequence to be encrypted.

Let X^n and Y^n represent the finite plaintext sequence $\{X_1,X_2,\ldots X_n\}$ and ciphertext sequence $\{Y_1,Y_2,\ldots Y_n\}$ respectively. It is assumed that each key Z is statistically independent from the plaintext sequence X^n for all n.

Cryptography is the key application field for message randomization [44]. This was first established by Shannon [38], who introduced the notion of the key-equivocation function f(*n*), which was defined to be the conditional entropy of the secret key given the first *n* digits of ciphertext. This is depicted as:

$$f(n) = H(Z|Y^n), \tag{4.4}$$

where Z represents the secret key and Y^n represents the first *n* digits of the ciphertext. Since f(n) can only decrease as n increases, Shannon called a cipher system ideal if f(n) approaches a non-zero value as n tends towards infinity [37]. The secret-key cipher system is considered strongly ideal if f(n) is constant as shown in , i.e.

$$H(Z|Y^n) = H(Z), \tag{4.5}$$

for all n. This implies that the ciphertext is statistically independent of the secret key [36]. A strongly ideal cryptosystem makes certain the secret key cannot be reconstructed by a cryptanalyst using a ciphertext-only attack. It was also noted [38] that if there existed an alphabet whose letters were equiprobable and independent, then a simple cipher would produce a strongly ideal cryptosystem regardless of the statistics for the secret key. Message randomization aims to simulate such an alphabet.

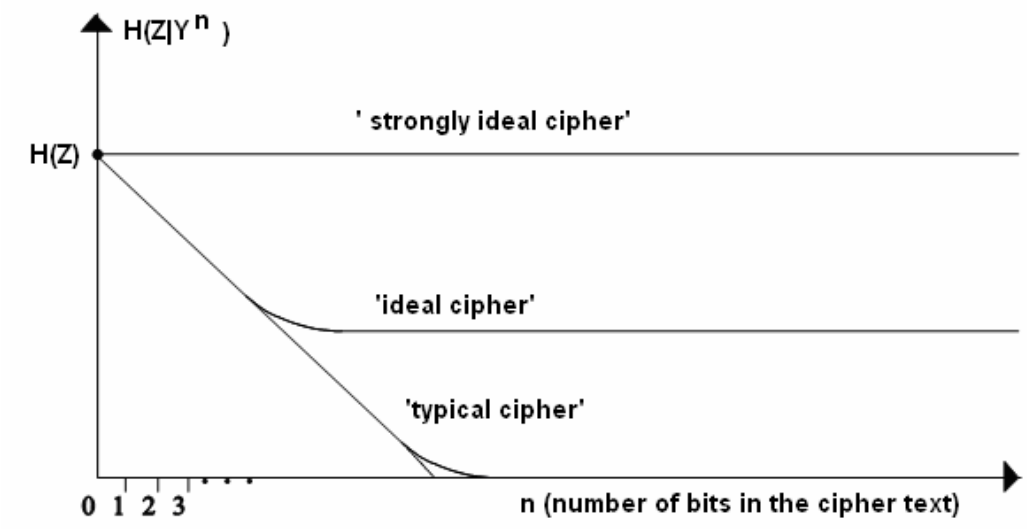

Figure 4.4 Illustration of the Key-Equivocation function f(n)

4.2.1. Key results obtained from the information-theoretic Homophonic coding analysis

In this section, some key results from the information-theoretic homophonic coding analysis that were proved in [37] are given:

- Proposition: If the plaintext sequence encrypted by means of a secret-key cipher is completely random, then the ciphertext sequence is also completely random, and is also statistically independent of the secret key.
- Corollary 1: If the plaintext sequence encrypted by means of a secret key cipher is completely random, then the cipher is strongly ideal, regardless of the probability distribution for the secret key.
- Corollary 2: If the plaintext sequence encrypted by a secret key cipher is completely random and all possible key values are equally likely, then the conditional entropy of the plaintext sequence, given the ciphertext sequence, satisfies

$$H(X^n|Y^n) = H(Z), \tag{4.6}$$

for all n sufficiently large. This implies in a ciphertext-only attack, the cryptanalyst can do no better than guessing at random from as many possibilities as there are possible

values of the secret key Z.

This illustrates the fact that virtually any secret-key cipher can be used as the cipher in a strongly ideal cipher system, provided that the plaintext source emits a completely random sequence. This is precisely what homophonic coding aims to achieve by converting non-uniformly distributed source into a completely random one.

4.2.2. Perfect & Optimum Homophonic Coding

A few definitions and proven propositions [37] clarifying what is classified as perfect & optimum homophonic coding is listed here:

- A homophonic coding scheme is called perfect if the resulting binary sequence is completely random.

- An optimum homophonic coder is defined [37] as a scheme minimizing the average codeword length, E[w], for a given source. The increase of the message length produced by a randomisation method may be measured by redundancy (r) which is defined as the difference between the average codeword length and the source entropy.

- A probability distribution is dyadic where there exists a distribution in which each probability is a negative integer power of 2 [47].

Consider the following propositions with homophonic channel output V, when the message source output U is used as input into the homophonic channel:

- The expression

$$H(U) \leq H(V) \leq E[W] = \sum_{i=1}^{M} p(v_i)\ l_i, \tag{4.7}$$

holds with equality on the left if and only if the homophonic channel is deterministic, and with equality on the right if and only if the homophonic coder is perfect. Moreover, there exists a binary arithmetic source coding of V such that the scheme is perfect if and only if $p(v)$ is dyadic for all possible values of v_i of V, where l_i is the length of the binary codeword assigned to v_i.

- A homophonic channel is optimum if and only if, for every u of U its homophones equal (in some order) the terms in the unique dyadic decomposition

$$P(U = u_i) = \sum_{j \geq 1} p_i^{(j)} \tag{4.8}$$

where the dyadic decomposition is either a finite of an infinite sum.

- For an optimum binary homophonic coder holds:

$$H(U) \leq H(V) = E[W] < H(U) + 2 \tag{4.9}$$

In order to choose among homophones, random bits are obtained form a table or a generator. If homophones have distinct choice probabilities for every source symbol (i.e. dyadic) [37], the redundancy does not exceed 2 bits and not more than four random bits on average are required to pick a homophone [36], [46]. There is a tradeoff between the number of redundant bits and the randomness achieved for the code sequence.

4.3. HOMOPHONIC CODING BASED ON ARITHMETIC CODING

The main aim of combining homophonic coding with arithmetic coding or even developing an algorithm for homophonic coding based on arithmetic coding is to randomize the source before it is encoded thus creating a strongly ideal cipher system.

The four main steps in the algorithm for homophonic coding based on arithmetic coding are [34], [48]:

1. Source modeling
2. Design of the homophonic channel
3. Random selection of the homophones
4. Arithmetic coding of the homophones

Source modeling involves estimating the statistics of the source in order to determine the sub interval width associated with each source symbol.

Design of the homophonic channel is to determine the homophones to be associated with each source symbol. This can be performed by decomposing the symbol probabilities dyadically. For example [45], [48], if a source symbol has a probability of $P(s_1) = 1/3$, the decomposition will occur as follow:

$P(s_1)=1/3$

$= 0.01010101...$ (binary)

$= 0.0101 + \epsilon$ (truncated)

$= 1/4 + 1/16 + \epsilon$

$= P(v_{11}) + P(v_{12}) + \epsilon$

The two (dyadic) homophones associated with s_1 are thus $v_{11} = 1/4$ and $v_{12} = 1/16$. A small error of ϵ will occur as a result of the truncation of the probabilities to finite precision. The augends or cumulative probabilities of the homophones represent the codewords for the homophones.

In the homophonic coding step, each symbol is mapped to one of its associated homophones. The homophones are normally selected by means of an external randomizer. It is this step in the coding process which introduces additional randomness into the message, thus causing the increase of entropy of a maximum of 2 bits/symbol [37].

The arithmetic encoding of homophones involves the encoding of each selected homophone by means of an arithmetic coding algorithm e.g. shift and add algorithm [45] etc. The output of the arithmetic coder is a string of binary digits. It is important to note that for this step, any implementation of arithmetic coding can be used [56]. Certain algorithms such as the shift and add algorithm perform faster than others [50].

4.3.1. Homophonic coding based on Arithmetic coding Example

This section uses an example to illustrate the above described algorithm; this particular example makes use of the shift-and-add implementation of arithmetic coding [52].

4.3.1.1. Source Modelling – Estimate source statistics

For the implementation of the shift-and-add homophonic coder, the source statistics are determined by counting the frequency of occurrence of each character in the message file. The validity of this approach is based on the assumption that the statistics remain stable and unchanged throughout the entire file.

After this the counts are scaled so that the total of all counts is less than 2^b where b is the word size (b is typically used as 16). This must be done to ensure that the augends fit into one word. The scaling operation may cause the counts of characters with low frequency to be 0. When this occurs the count is set equal to 1.

Suppose we want to encode the following message:

aabcc badd abad

This message consists off 5 different characters. The total character count is 15. Table 4.2 shows the statistics for the message.

Table 4.2 Statistics for the string *aabaa badd abad*

Character	Frequency	Probability Freq./total
a	5	0.3333
b	3	0.2
c	2	0.1333
d	3	0.2
space	2	0.1333

4.3.1.2. Design of the Homophonic channel – Determination of the homophones

Once the statistics of the source have been estimated, the probability of each source symbol has to be decomposed and approximated by dyadic homophones. In this way the homophonic channel is formed. In general, a small error will always occur when approximating a probability dyadically.

Suppose for example that the probability for character *d* is 0.2 and that the word size is 4 (b=4). The dyadic decomposition for this probability is illustrated in Table 4.3. Note that there are two homophones for character *d*, namely 1/8 and 1/16.

Table 4.3 Dyadic decomposition of 0.2 with a word size of 4.

Decimal			Binary		
Prob.	**Dyadic Decomp.**	**Result**	**Prob.**	**Dyadic Decomp.**	**Result**
0.2	1/8	0.1875 + ε	0.00110011	0.001	0.0011 + ε
	1/16			0.0001	
	ε			ε	

The size of the error made can be decreased by increasing the word size. Suppose that the word size was 8. Then the dyadic composition for character *d* would have been as illustrated in Table 4.4.

Table 4.4 Dyadic decomposition of 0.2 with a word size of 8

Decimal			Binary		
Prob.	**Dyadic Decomp.**	**Result**	**Prob.**	**Dyadic Decomp.**	**Result**
0.2	1/8	0.199 + ε	0.00110011	0.001	0.00110011 + ε
	1/16			0.0001	
	1/128			0.0000001	
	1/256			0.00000001	
	ε			ε	

The price paid for this reduction in error are that there are more homophones and thus a bigger symbol alphabet.

The number of homophones for each character are counted and used later when mapping the characters to symbols. The array that stored the counts is "expanded" into a new array containing the homophones. The source alphabet is expanded from 5 elements to 8 elements, as illustrated in Table 4.5. Note that the homophone with the largest probability for a character has the smallest symbol number corresponding with that character. In the Table below it can be seen how each character is decomposed into its respective homophones.

Table 4.5 Statistics of the homophones for the source message *aabcc badd abad*

Characters			**Homophones**				**Cumulative Probabilities**	
Number	**Character**	**Prob.**	**Number**	**Prob.**	**Approx.**	**Prob. (binary)**	**Actual (binary)**	**Integer represen-tation**
1	a	0.3333	1	1/4	0.3125	.01	.0000	0
			2	1/16		.0001	.0100	4
2	b	0.2	3	1/8	0.1875	.001	.0101	5
			4	1/16		.0001	.0111	7
3	c	0.1333	5	1/8	0.125	.001	.1000	8
4	d	0.2	6	1/8	0.1875	.001	.1010	10
			7	1/16		.0001	.1100	12
5	space	0.1333	8	1/8	0.125	.001	.1101	13

This decomposition of the symbol probabilities into their respective homophones is depicted in Figure 4.6.

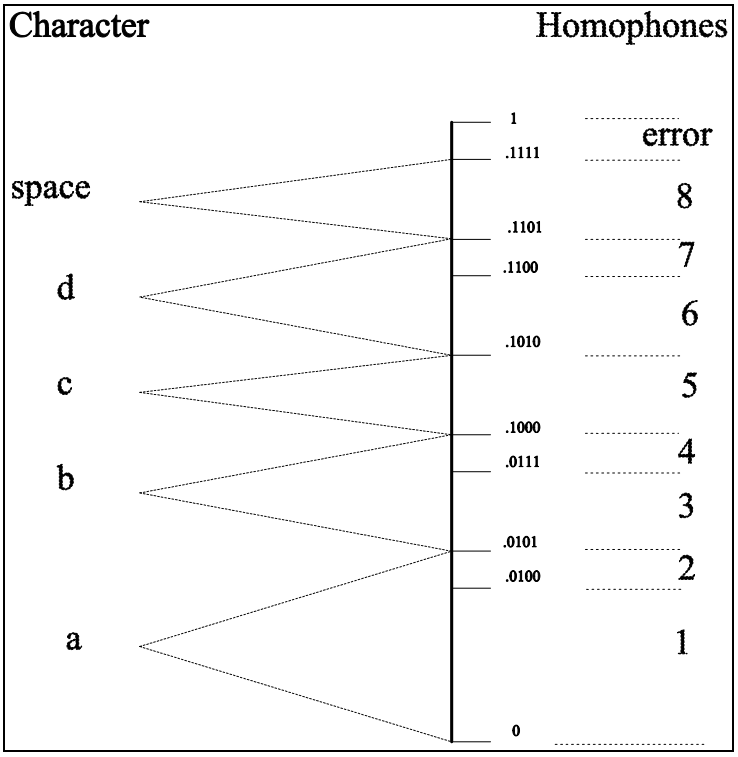

Figure 4.6 Mapping of the characters to their respective homophones

Next we need to set up a table for mapping the characters to their homophones, or symbols. In the char_to_symbol array, only the symbol number of the characters first homophone is stored. This is depicted in Table 4.6. The number of homophones and a random generator is later used to decide the offset from that position.

Table 4.6 Char_to_symbol array

Char	a	b	c	d	space
Char_to_symbol	1	3	5	6	8

The decoder needs to convert symbols to characters. A symbol_to_char array is therefore

needed. Let n be the number of homophones of character c. In the symbol_to_char array, n consecutive places are filled with character c. Take for example character *a*. This character has 2 homophones. The first 2 places in the Symbol_to_char array is therefore filled with *a*'s. The next character *b* also has 2 homophones and will therefore be filled into the next 2 places of the Symbol_to_char array. This is depicted in Table 4.7.

Table 4.7 Symbol_to_char array

Symbol	1	2	3	4	5	6	7	8
Symbol_to_char	a	a	b	b	c	d	d	space

4.3.1.3. Homophonic Coding- Selection of homophones

For the homophonic coding of a given string of source symbols, each symbol is mapped into one of its homophones. The choice of which homophone to select is done by means of an external random number generator in this example. In this way additional randomness is introduced into the message source symbols, and it is this additional randomness which results in an increase of entropy (of maximally 2 bits/symbol). The output of the arithmetic coder is thus a string of binary digits.

To determine which homophone to choose, the following steps are perfomed:

1. Determine the number of homophones for the input character.

2. Get the frequency count for the character and chooses a random number smaller than the count value.

3. Sum the number of homophones minus one and the value listed in the Char_to_symbol array to determine the symbol number of the smallest homophone for the input character.

4. Sum homophones, from the smallest to the largest, until the sum is larger than the random number. The last homophone summed is the homophone to be encoded. The symbol number for this homophone is calculated by adding the homophone number to the value in the Char_to_symbol array.

Consider the example input string *aabcc badd abad*. Note that character *d* has 2 homophones namely 2 and 1. They are stored as symbols 6 and 7 respectively. In the Char_to_symbol array the value 6 is listed for *d*. This value plus the number of homophones minus one is: 6+(2-1)=7. This value corresponds with the symbol number for *d*'s smallest homophone.

The random generator chooses a number between 0 and 3 (the count for *d* is 3). If it is less than 1, symbol 7 is chosen. If not, symbol 6 is chosen. Homophones are summed from the smallest to the largest. When this sum is larger than the random number, the last homophone summed is chosen as the symbol. Say for example that the random generator gives the number 2.3. The homophones for *d* is 1/8 and 1/16. Remember that this is represented as 2 and 1 in the implementation of the shift-and-add arithmetic coder. Because 2.3 is larger than 1, the two homophones are summed to obtain 3. This is larger than 2.3 and the homophone 1/8 is chosen as the symbol to be encoded.

4.3.1.4. Arithmetic coding- Encode the selected homophone

The homophonic channel is followed by the arithmetic source coder. Any implementation of arithmetic coding would be suitable for this algorithm.

For this case, consider the encoding of the first 4 characters of the source character string *aabcc badd abad*. Suppose that after homophone selection the symbol string is *2,1,3,5*. The encoding process is illustrated in Figure 4.8.

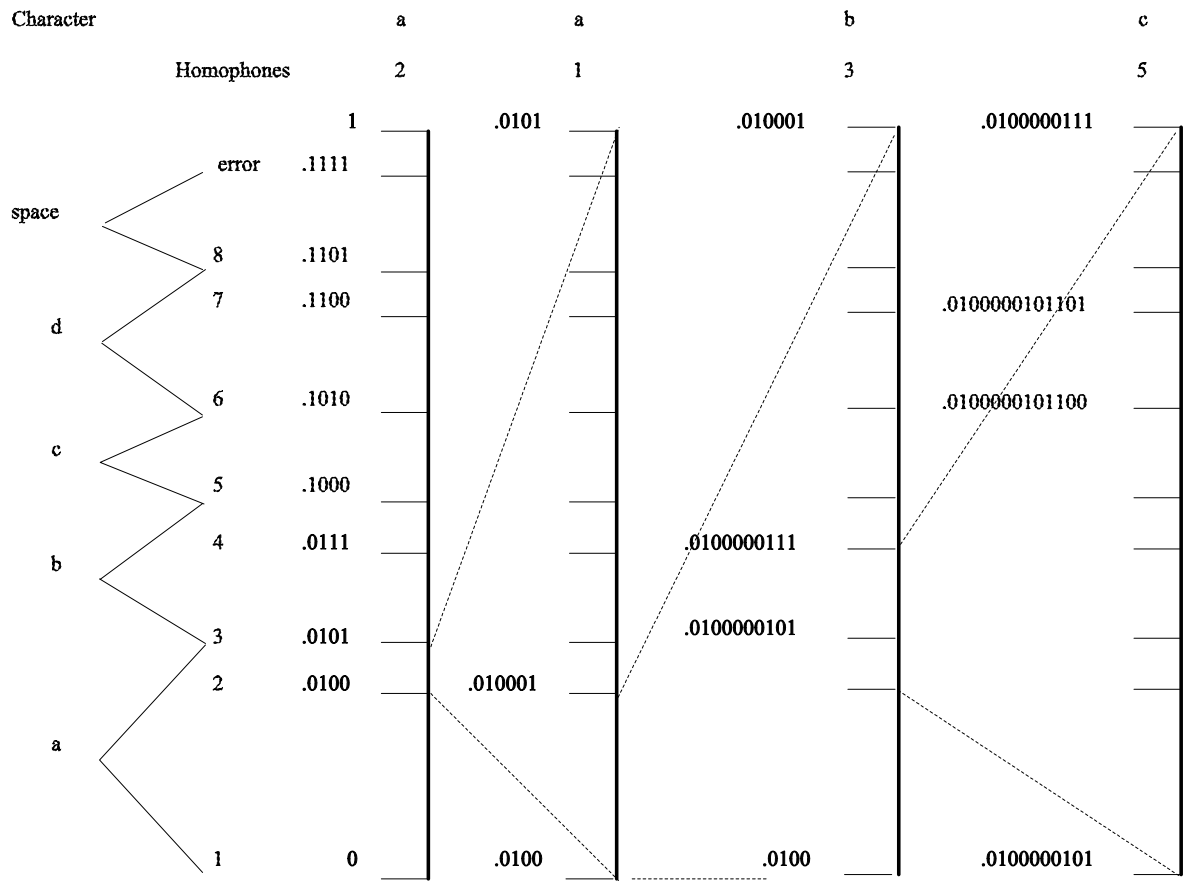

Figure 4.8 Encoding of the string *aabc*

4.3.1.5. Decoder Operation

At the decoder the first 4 bits of the string are compared with the values of the augends. The decoder now maps the symbol to the corresponding character by using the symbol_to_char array. The character *a* is then written to the output file. After this has been done, the augend for symbol 4 is subtracted, the code string is shifted left and filled with the contents of the buffer as described earlier.

This concludes the example of the homophonic coder based on arithmetic coding.

4.4. SUMMARY

The two main coding techniques explored here are Homophonic and Arithmetic coding. These two coding techniques form the underlying pillars of the algorithm implemented in this dissertation. An in-depth investigation of the arithmetic and homophonic coding schemes has been given in this chapter along with examples to illustrate how each of the coding schemes work. The pros and cons of choosing Arithmetic coding as the entropy coder for the JPEG compression and to combine with Homophonic coding is explored.

The last section of this chapter focuses on homophonic coding based on arithmetic coding as this concept/method is used in the algorithm developed and implemented to embed a watermark in a JPEG compressed image as will be detailed in the following chapter. It can be seen that the homophonic coding algorithm based on arithmetic coding thus achieves to map a non-uniformly distributed sequence of source symbols into a uniformly distributed sequence of bits which are then encoded using the source coding technique [51] of arithmetic coding.

CHAPTER 5. IMPLEMENTATION

This chapter forms the core of this dissertation and highlights the algorithm that has been designed and implemented in order to create a unique watermarking encoding scheme for compressed material such JPEG images. The implementation approach, design alternatives and implementation choices are explored. The results from this implementation are documented in Chapter 6 of this dissertation.

5.1. IMPLEMENTATION OVERVIEW

The main aim of this dissertation is develop an algorithm to combine homophonic coding and arithmetic coding in order to embed a watermark into compressed material, in this case JPEG images [58],[59].

This algorithm was implemented in software using the Microsoft .NET 2003 development platform. The algorithm was developed using Visual C++ within this environment.

The approach taken for this implementation was to first ensure that an 8-bit grayscale bitmap image could be read and stored as an 8-bit grayscale JPEG image. Thereafter arithmetic encoding was introduced into the JPEG standard [54], [62] as this is not normally supported. Once an arithmetically encoded JPEG image could be successfully created using this algorithm, homophonic coding was introduced. This enabled the implementation of an algorithm for homophonic coding based on arithmetic coding within the JPEG standard. Instead of using an external randomizer for the selection of the homophones, a bitstring containing proprietary information was used, thus intrinsically embedding a watermark which is then encoded using arithmetic coding.

5.2. JPEG

For the implementation of the JPEG component of the algorithm, the JPEG Library [54] was utilized. In order to modify the JPEG library to adapt to the design of the proposed algorithm in this dissertation, it is imperative to understand the JPEG file layout and format [60] as is investigated in following sections.

5.2.1.1. JPEG File Layout

A JPEG file is partitioned by markers. Each marker is immediately preceded by an all 1 byte (0xff). Although there are more markers, Table 5.1 highlights the most important markers.

Table 5.1 JPEG file markers

Marker Name	Marker Identifier	Description
SOI	0xd8	Start of Image
APP0	0xe0	JFIF application segment
APPn	0xe1 – 0xef	Other APP segments
DQT	0xdb	Quantization Table
SOF0	0xc0	Start of Frame
DHT	0xc4	Huffman Table
SOS	0xda	Start of Scan
EOI	0xd9	End of Image

If a 0xff byte occurs in the compressed image data either a zero byte (0x00) or a marker identifier follows it. Normally the only marker that should be found once the image data is started is an EOI. When a 0xff byte is found followed by a zero byte (0x00) the zero byte must be discarded.

A JPEG file consists of the following 8 parts:

1. A Start of Image SOI

2. An APP0 Marker
    - APP0 length
    - Identifier
    - Version
    - Units for X & Y densities
    - X density
    - Y density
    - Thumbnail horizontal pixels
    - Thumbnail vertical pixels
    - Thumbnail RGB bitmap
3. APPn Markers where n can be from 1 to 15 (Optional)
    - APPn length
    - Application specific information
4. One or more quantization tables DQT
    - Quantization table length
    - Quantization table number
    - Quantization table
5. A Start of Frame SOF0
    - Start of Frame length
    - Precision (Bits per pixel per colour component)
    - Image height
    - Image width
    - Number of colour components
    - For each component
        - An ID
        - A vertical sample factor
        - A horizontal sample factor
        - A quantization table#
6. One or more Huffman tables DHT
    - Huffman table length

- Type, AC or DC
- Index
- A Bits table
- A Value table

7. A Start of Scan SOS
   - Start of Scan length
   - Number of colour components
   - For each component
     - An ID
     - An AC table#
     - A DC table#
   - Compressed image data (Not included in Start of Scan length)
8. An End of Image EOI

5.2.1.2. JPEG File Format

Before exploring procedural details, it is helpful to understand the image data format that the JPEG library expects or returns. The standard input image format is a rectangular array of pixels, with each pixel having the same number of "component" or "sample" values (colour channels). It is required to specify how many components there are and the colour space interpretation of the components. Most applications will use RGB data (three components per pixel) or grayscale data (one component per pixel). In this implementation, grayscale data is considered.

Pixels are stored by scanlines, with each scanline running from left to right. The component values for each pixel are adjacent in the row. A 2-D array of pixels is formed by making a list of pointers to the starts of scanlines; so the scanlines need not be physically adjacent in memory. Even if you process just one scanline at a time, you must make a one-element pointer array to conform to this structure. Scanlines are always processed top-to-bottom. An entire image can be processed in one call if it is all in memory, but usually it's simplest to

process one scanline at a time.

It is also important to have an idea of the structure of the JPEG file format as this gives the user an idea of what the markers entail and how these are populated depending on whether arithmetic or Huffman coding is used as the entropy coder of choice.

5.2.1.3. Independent JPEG Group Library (IJG)

In order to read in an 8-bit grayscale bitmap and compress this into a 8-bit grayscale JPEG image the IJG library [54] was utilized in this algorithm. The distribution used is the sixth public release of the Independent JPEG Group's free JPEG software.

The IJG distribution contains of subroutine libraries for JPEG compression and decompression.

The conversion programs used are cjpeg and djpeg where cjpeg compresses an image file into JPEG format and djpeg decompresses a JPEG file back into a conventional image format.

The IJG library is capable of supporting all JPEG baseline, extended sequential, and progressive DCT processes. Arithmetic coding has previously not been supported due to legal reasons pertaining to patents around the Q-coder implementation of the arithmetic coder.

5.3. ARITHMETIC ENCODER

Arithmetic coding is a state-of-the-art lossless entropy data compression method which offers better compression performance than the well-established Huffman entropy coding process. The JPEG standard specifies a particular arithmetic coding scheme to be used optionally as alternative to Huffman coding.

In the light of not being able to implement the Q-coder algorithm, the following arithmetic

coding algorithms were considered:

- Shift-and-Add algorithm
- JPEG standard algorithm implementation

For this implementation, it was decided to make use of the algorithm as described in the JPEG standard and the code implemented independently using the JPEG standard and can be freely used. This implementation was chosen for ease of integration with the IJG Library.

The arithmetic coding implementation as developed by G. Vollbeiding [65] is an arithmetic encoding and decoding back-end for JPEG as specified in [24].

In order to incorporate this arithmetic encoder with the Jpeg Library in order to produce arithmetically encoded JPEG images, parameters in certain files such as Jcparam.c were changed. Also jcarith for compression, jdarith for decompression and jaricom were included in the Jpeg Library.

5.4. WATERMARK ENCODING ALGORITHM

In this section a clear description of the proposed watermark encoding scheme for JPEG images is given.

5.4.1.1. Algorithm Overview

The algorithm utilizes the concept of combining Homophonic and Arithmetic coding in order to embed a watermark intrinsically within the message stream of a JPEG compressed image.

The four main steps in the algorithm as described in Section 4.3 are:

- Source modeling
- Design of the homophonic channel
- Selection of the homophones

- Arithmetic coding of the homophones

We will now explore these steps for this specific implementation incorporated with the JPEG standard.

5.4.1.2. Source Modelling - Probability Estimation

The probability estimation state machine for the arithmetic encoder consists of a number of sequences of probability estimates. These sequences are interlinked in a manner which provides probability estimates based on approximate symbol counts derived from the arithmetic coder renormalization. Some of these sequences are used during the initial "learning" stages of probability estimation; the rest are used for "steady state" estimation.

In the partitioning of the current probability interval into two sub-intervals, the sub-interval for the less probable symbol (LPS) and the sub-interval for the more probable symbol (MPS) are ordered such that usually the MPS sub-interval is closer to zero. Therefore, when the LPS is coded, the MPS sub-interval size is added to the bit stream. This coding convention requires that symbols be recognized as either MPS or LPS rather than 0 or 1. Qe represents the LPS probability estimate and Qe(S) LPS probability estimate for context index S.

Each entry in the probability estimation state machine is assigned an index, and each index has associated with it a Qe value and two Next_Index values. The Next_Index_MPS gives the index to the new probability estimate after an MPS renormalization; the Next_Index_LPS gives the index to the new probability estimate after an LPS renormalization. Note that both the index to the estimation state machine and the sense of the MPS are kept for each context-index S. The sense of the MPS is changed whenever the entry in the Switch_MPS is one.The Qe values listed in are expressed as hexadecimal integers. To approximately convert the 15-bit integer representation of Qe to a decimal probability, divide the Qe values by 4/3.

Table 5.2 Qe values and probability estimation state machine

Index	Qe _Value	Next_ Index _LPS	Next_ Index _MPS	Switch _MPS	Index	Qe _Value	Next_ Index _LPS	Next_ Index _MPS	Switch _MPS
0	X'5A1D'	1	1	1	57	X'01A4'	55	58	0
1	X'2586'	14	2	0	58	X'0160'	56	59	0
2	X'1114'	16	3	0	59	X'0125'	57	60	0
3	X'080B'	18	4	0	60	X'00F6'	58	61	0
4	X'03D8'	20	5	0	61	X'00CB'	59	62	0
5	X'01DA'	23	6	0	62	X'00AB'	61	63	0
6	X'00E5'	25	7	0	63	X'008F'	61	32	0
7	X'006F'	28	8	0	64	X'5B12'	65	65	1
8	X'0036'	30	9	0	65	X'4D04'	80	66	0
9	X'001A'	33	10	0	66	X'412C'	81	67	0
10	X'000D'	35	11	0	67	X'37D8'	82	68	0
11	X'0006'	9	12	0	68	X'2FE8'	83	69	0
12	X'0003'	10	13	0	69	X'293C'	84	70	0
13	X'0001'	12	13	0	70	X'2379'	86	71	0
14	X'5A7F'	15	15	1	71	X'1EDF'	87	72	0
15	X'3F25'	36	16	0	72	X'1AA9'	87	73	0
16	X'2CF2'	38	17	0	73	X'174E'	72	74	0
17	X'207C'	39	18	0	74	X'1424'	72	75	0
18	X'17B9'	40	19	0	75	X'119C'	74	76	0
19	X'1182'	42	20	0	76	X'0F6B'	74	77	0
20	X'0CEF'	43	21	0	77	X'0D51'	75	78	0
21	X'09A1'	45	22	0	78	X'0BB6'	77	79	0
22	X'072F'	46	23	0	79	X'0A40'	77	48	0
23	X'055C'	48	24	0	80	X'5832'	80	81	1
24	X'0406'	49	25	0	81	X'4D1C'	88	82	0
25	X'0303'	51	26	0	82	X'438E'	89	83	0
26	X'0240'	52	27	0	83	X'3BDD'	90	84	0
27	X'01B1'	54	28	0	84	X'34EE'	91	85	0
28	X'0144'	56	29	0	85	X'2EAE'	92	86	0
29	X'00F5'	57	30	0	86	X'299A'	93	87	0
30	X'00B7'	59	31	0	87	X'2516'	86	71	0
31	X'008A'	60	32	0	88	X'5570'	88	89	1
32	X'0068'	62	33	0	89	X'4CA9'	95	90	0
33	X'004E'	63	34	0	90	X'44D9'	96	91	0
34	X'003B'	32	35	0	91	X'3E22'	97	92	0
35	X'002C'	33	9	0	92	X'3824'	99	93	0
36	X'5AE1'	37	37	1	93	X'32B4'	99	94	0
37	X'484C'	64	38	0	94	X'2E17'	93	86	0
38	X'3A0D'	65	39	0	95	X'56A8'	95	96	1
39	X'2EF1'	67	40	0	96	X'4F46'	101	97	0
40	X'261F'	68	41	0	97	X'47E5'	102	98	0
41	X'1F33'	69	42	0	98	X'41CF'	103	99	0
42	X'19A8'	70	43	0	99	X'3C3D'	104	100	0
43	X'1518'	72	44	0	100	X'375E'	99	93	0
44	X'1177'	73	45	0	101	X'5231'	105	102	0
45	X'0E74'	74	46	0	102	X'4C0F'	106	103	0
46	X'0BFB'	75	47	0	103	X'4639'	107	104	0
47	X'09F8'	77	48	0	104	X'415E'	103	99	0
48	X'0861'	78	49	0	105	X'5627'	105	106	1
49	X'0706'	79	50	0	106	X'50E7'	108	107	0
50	X'05CD'	48	51	0	107	X'4B85'	109	103	0
51	X'04DE'	50	52	0	108	X'5597'	110	109	0
52	X'040F'	50	53	0	109	X'504F'	111	107	0
53	X'0363'	51	54	0	110	X'5A10'	110	111	1
54	X'02D4'	52	55	0	111	X'5522'	112	109	0
55	X'025C'	53	56	0	112	X'59EB'	112	111	1
56	X'01F8'	54	57	0					

5.4.1.3. Design of the homophonic channel

As previously described, once the statistics of the source have been estimated, the probability of each source symbol has to be decomposed and approximated by dyadic homophones. In this way the homophonic channel is formed. The number of homophones for each character are counted and used later when mapping the characters to symbols.

The code for the determination of homophones is:

```
number_homophones[0] = 0;
number = 1;
hx = 0;
for(hy=0;hy<256;hy++)
{
 count = Stat[hy];
 power_count = 1;
 homophon = pow(2,bufsize-1);   /* biggest homophone */
 while(count != 0)
 {
       while((test=homophon) > count)  /* compare homophone and count*/
  {
        homophon >>= 1;    /*  divide by 2  */
        power_count++;         /* increase probability length */
  }
  homo[number++] = homophon;      /* fill in homophones array  */
  Prob[number-1] = power_count;       /* fill in probability array */
  hx++;
  count -= test;
 }
 number_homophones[hy+1] = hx;      /*  Note number of homophones */
```

```
    if(Stat[hy] == 0) number++;
  }
```

In order to setup the table for mapping the characters to their homophones, or symbols, the following array is required. In the char_to_symbol array, only the symbol number of the characters first homophone is stored. The number of homophones and a watermark bitstring is later used to decide the offset from that position.

The char_to_symbol array is filled in with the following code:

```
number2 = 0;
for(hy=0;hy<256;hy++)
{
 char_to_symbol[hy] = ++number2 + number_homophones[hy];
 if((number_homophones[hy+1]-number_homophones[hy])>0) number2--;
}
```

The decoder needs to convert symbols to characters. A symbol_to_char array is therefore needed.

The Symbol_to_char array is filled in with the following code:

```
number = 1;
for(hy=0;hy<256;hy++)
{
  n = (number_homophones[hy+1]-number_homophones[hy]);
  if(n > 1)   for(hx=0;hx<n;hx++)  symbol_to_char[number++] = hy;
    else  symbol_to_char[number++] = hy;
}
```

```
symbol_to_char[number] = ++number;
```

5.4.1.4. Homophonic coding - Selection of Homophones- Watermark Embedding

For the homophonic coding of a given string of source symbols, each symbol is mapped into one of its homophones.

The choice of which homophone to select is done by means of obtaining a number from a bitstring, containing watermark information such as the author of the image and the date created, instead of using a random number generator. In this way additional information is introduced into the message source symbols.

To determine which homophone to choose, the following steps are performed:

- Determine the number of homophones for the input character.
- Get the frequency count for the character and chooses the next sequential bit in the watermark bitstring. A check is performed to ensure that it is a number smaller than the count value else the number is reduced by 1.
- Sum the number of homophones minus one and the value listed in the Char_to_symbol array to determine the symbol number of the smallest homophone for the input character.
- Sum homophones, from the smallest to the largest, until the sum is larger than the watermark symbol. The last homophone summed is the homophone to be encoded. The symbol number for this homophone is calculated by adding the homophone number to the value in the Char_to_symbol array.

5.4.1.5. Arithmetic coding of homophones

The homophonic channel is followed by the arithmetic source coder. The arithmetic coding algorithm used in this dissertation follows from the implementation developed from the JPEG standard as detailed in Section 5.3

5.4.1.6. Decoder Operation

The decoder operation completes this process and a JPEG image containing an intrinsically embedded watermark is formed

5.5. SUMMARY

This chapter entails the algorithmic description of the watermarking encoding scheme proposed and implemented in this dissertation. It can be seen from this implementation that the underlying principles of homophonic coding based on arithmetic coding form the basis of this algorithm.

This watermark encoding scheme implemented here makes use of the watermark information to choose the homophones which are then encoded using arithmetic coding. This implies that the watermark information is intrinsically embedded in the message stream using encoding techniques thus making it difficult for a malicious user to extract and eliminate the watermark information.

This encoding scheme is incorporated with the JPEG algorithm thus allowing for the watermarking of JPEG compressed images.

CHAPTER 6. RESULTS

This chapter covers the results from the implementation of the watermark encoding algorithm implementation as discussed in Chapter 5. The main outcomes of this implementation include a unique watermarking scheme with enhanced security aspects for compressed JPEG images, enhanced compression performance in comparison to Huffman coding and minimal distortion to quality of image.

6.1. JPEG COMPRESSION

Digital watermarking techniques known as perceptually based watermarks are designed to exploit aspects of the human visual system in order to provide an imperceptible, yet robust watermark [67].

In using Arithmetic coding instead of Huffman coding, it can be seen that there is a significant increase in compression of the JPEG image. An example file is shown using a 512 x 512 pixel image of Lena.

	Lena.bmp original image	Lena.jpg Huffman Coding	Lena.jpg Arithmetic Coding
Image Size (Bytes)	263 222	32 664	30 870

Table 6.1 Comparison of file sizes using different entropy encoders for JPEG compression

Table 6.1 gives an indication of the reduction in image size using the different entropy encoders, namely Huffman coding and Arithmetic coding. This shows that arithmetic coding provides a more efficient compression factor.

In investigations of the entropy per symbol once the watermark information has been embedded using the homophonic encoding with arithmetic coding, it was found that the entropy does not exceed the source entropy by more than 2 bits. The entropies achieved were all < 2bits/per symbol in increase when compared to the entropy/symbol for the source code.

6.2. SECURTY ASPECTS

In the design of a watermarking algorithm, it needs to be taken into consideration that watermarking techniques are subject to attacks [71]. In using a homophonic algorithm combined with arithmetic coding, protection against certain attacks is provided.

Cipher text-only attacks imply that the encryption algorithm and the ciphertext to be decoded is known to the cryptanalyst. A known plaintext attack is when the encryption algorithm, ciphertext to be decoded and one or more plaintext-ciphertext pairs are formed with the secret key.

Homophonic coding provides protection against ciphertext-only attacks and chosen/known-plaintext attacks. Source coding, on the other hand protects against ciphertext-only attacks.

As explored in Section 4.2.1, homophonic coding can be used to create a strongly ideal cipher-key system. As per the proposition: If the plaintext sequence encrypted by means of a secret key cipher is completely random, then the cipher is strongly ideal, regardless of the probability distribution for the secret key.

In this implementation, the watermark bitstring is used as an ‘external’ key when choosing the homophones and is embedded intrinsically into the message system. This shows that the watermark encoding algorithm implemented here can be seen as a ‘key-less’ encryption technique.

6.3. WATERMARK ENCODING SCHEME: RESULTING IMAGES

Having chosen arithmetic coding as the entropy encoder for this implementation, a particular image reader was required in order to view the arithmetically encoded JPEG images as most of the commonly-used image viewers such as MS Paint and Adobe Photoshop do not support

arithmetic coding.

The "Jpegcrop" program [64] was used to view the JPEG images that used arithmetic encoding. Jpegcrop uses the original transformation support code of the IJG software, with direct function calls.
The following shows a sample of an image before and after it has been processed through the newly developed watermark encoding scheme.

Figure 6.1 Lena 8-bit bitmap image

Figure 6.1 indicates the original bitmap image which is used as input into the JPEG watermark encoding algorithm.

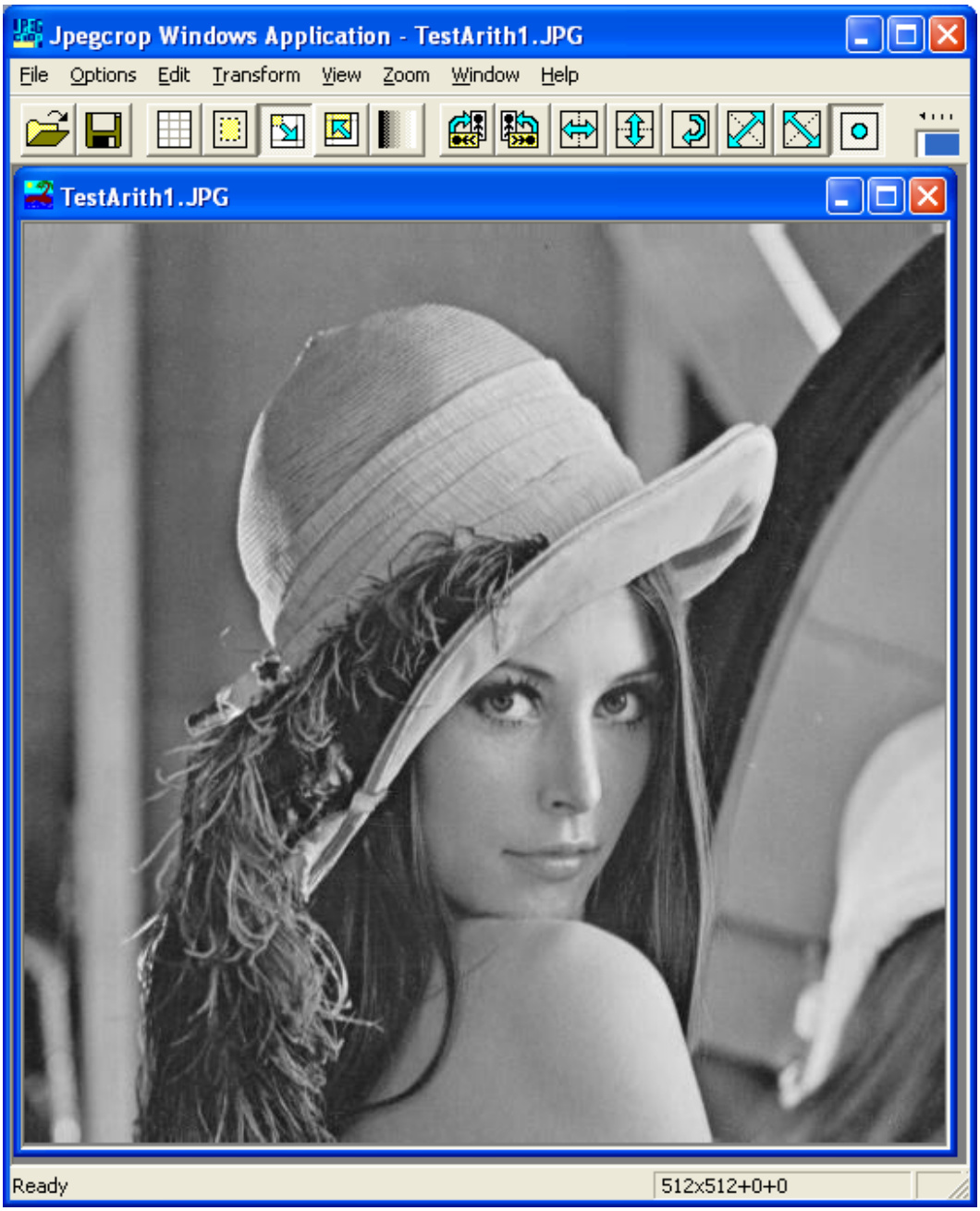

Figure 6.2 Lena JPEG image

In the above figure, the Lena image has been arithmetically encoded and a watermark bit string has been intrinsically embedded. The Jpegcrop program is used to view these images. It can be seen that there is almost no distortion to be detected between the original and

watermarked image.

The following table shows some of the PSNR values achieved for certain test images which are all grayscale images.

	Lena.jpg 512x512 pixels	**GoldHill.jpg 351x352 pixels**	**Lena.jpg 169x160 pixels**	**Peppers.jpg 100x100 pixels**
PSNR	36.37dB	39.36 dB	41.45 dB	45.87 dB

Table 6.2 PSNR values of various images watermarked using the new watermark encoding scheme

From the above it can be seen that this encoding scheme introduces minimal distortion as all the PSNR values are above the accepted norm of 35dB.

6.4. WATERMARK DETECTION

In order to determine the watermark that was embedded into the image, the reverse operation of the newly developed algorithm is performed on the watermarked image. Using the original image with this operation enables one to establish which homophones were chosen and thus the bitstring, containing copyright information, can also be determined. By determining the bitstring, it can thus be established as to what the embedded watermark was. Examples of detected watermarks would be bitstrings containing copyright information such as the name of the author, time stamp etc.

To be able to determine whether an image was copied, the image can be run through the newly developed algorithm to detect if it contains a watermark as explained above. If the image contains the watermark, it is obvious that the image was copied form the original JPEG.

At this stage of algorithmic development, once the watermark has been embedded, it is fairly robust and extraction of the watermark is not possible currently but can be explored further. The reason for this is that the watermark is intrinsically embedded as part of the encoding scheme and in order to extract the watermark, one would need to first complete the decoding operation on the image and then perform the encoding operation again to result in an image which does not contain the original watermark. Hence to physically extract the watermark it would be required to run the image through the newly developed algorithm which should only be available to the rightful owners of the JPEG images.

6.5. SUMMARY

Different data hiding techniques [70] exist, and even though other entropy based embedding techniques have been employed [63], the technique presented in this dissertation is unique and presents a completely different approach using the concept of combining two coding schemes.

In doing so, a unique watermark encoding scheme has been established for grayscale Jpeg images which embeds a watermark into the message stream by means of choosing the homophones by means of a watermark bitstring. It is also noted that any watermarking scheme is prone to attacks, by making use of homophonic coding, this watermarking scheme provides protection against cipher-text only and known-plaintext attacks.

It was also noted that arithmetic coding provides slightly better compression in comparison to Huffman coding when used for JPEG compression.

This implementation utilizes homophonic coding in such a manner that the entropy/symbol increase is within the expected theoretical prediction of 2 bits per symbol.

CHAPTER 7. CONCLUSION AND FUTURE WORK

The main purpose of this dissertation is to develop a watermark encoding scheme for JPEG compressed images such that the scheme does not distort the image although it may slightly expand the size of the image. This watermark technique involves the use of entropy coding in specific arithmetic coding and substitution technique, namely homophonic coding.

7.1. SUMMARY

An algorithm combining homophonic coding and arithmetic coding within the JPEG standard has been developed. It could be seen that arithmetic coding employed on its own yielded JPEG images in the region of 1 - 1.5 % decrease in image sizes when compared with an image that made use of Huffman coding as the entropy coder of choice for 8-bit grayscale images. This clearly indicates one of the reasons for arithmetic coding being the entropy encoder of choice for this particular algorithm.

In combining both homophonic coding with arithmetic coding when compressing images using the JPEG standard, the following was achieved:

- Homophonic coding can be used to embed a watermark in a JPEG image using the watermark information for the selection of the homophones.
- The proposed algorithm is viewed as a ‘key-less’ encryption technique, taking into consideration the security aspect introduced by the message randomisation property of homophonic coding, where an external bitstring is used as a ‘key’ and is embedded intrinsically into the message stream.
- Grayscale JPEG image with distortion with PSNR values of greater than 35dB
- In utilizing homophonic coding with arithmetic coding, the resulting increase in the

entropy of the images is within the expected increase of 2 bits per symbol.

7.2. CONTRIBUTION

Digital Rights Management (DRM) has been used as the umbrella term referring to any of several technologies used to enforce policies for controlling access to digital data and hardware. DRM technically handles the description, layering, analysis, monitoring and enforcement of usage restrictions that accompany a specific instance of digital material. The DRM technique which has been the focus in this research is that of a watermarking technique [72] for digital images in specific for JPEG compressed images.

In implementing this algorithm, it has been identified that a tradeoff is made regarding the size of the image, as there is an increase in size for the resulting images, and the additional security provided by 'message randomisation' introduced by the homophonic coding element. In spite of this tradeoff, it can be seen that this is a unique watermarking technique which utilizes encoding methods thus making the removal of the intrinsically embedded watermark more difficult to remove without significantly degrading the quality of the image.

The watermark can be detected by performing the reverse algorithmic operation. The one drawback in watermark detection is that the original image is required in order to establish the information held by the watermark.

The newly developed watermark encoding algorithm accomplishes the following:

- Minimal distortion of image achieved;
- Better compression ratios with use of arithmetic coding;
- Watermark encoding by use of homophonic coding within expected entropy increase of < 2 bits per symbol in comparison to the original image; and
- Enhanced security by use of homophonic coding providing protection against:
    - Ciphertext-only attacks
    - Chosen/known plaintext attacks.

7.3. FUTURE WORK

This research has thus formed the basis for further research to be performed. This would be in the area of expanding on the digital material on which this watermark technique can be applied to. This research can be further extended to material such as MP3 and MPEG material that make use of some entropy encoder. This would enable the copyright protection of this kind of digital material by applying the underlying principles of the algorithm presented in this dissertation to the appropriate digital material.

Other areas of research that stem from this dissertation can be an investigation of how homophonic coding combined with Huffman coding can be utilized for the purpose of embedding a watermark. The design and development of such an algorithm and the comparison thereof with the algorithm utilizing homophonic coding and arithmetic coding such as the one investigated in this dissertation, is left as a potential research project in this field.

It can be seen from this research investigation that a new avenue of using encoding techniques can be employed for the use in the field of digital watermarking with the resulting aim of providing copyright protection for digital material especially material that is utilized on the Internet. This work serves as a contribution in the field of Digital Rights Management and can be used as a platform for further research in this area of interest.

REFERENCES

[1] R.Oppliger, *"Security Technologies for the World Wide Web," in Intellectual Property Protection*, 2nd ed Norwood, MA: Artech House, 2003, pp. 347-357.

[2] G.P.Schneider and J.T.Perry, *Electronic Commerce*, 2nd ed Canada: Course Technology Thomson Learning, 2001, p.198.

[3] H. Berghel and L. O'Gorman, "Protecting Ownership Rights through Digital Watermarking," *IEEE Computer*, vol. 29, no.7, pp. 101-103, 1996.

[4] W.T. Penzhorn and W.C. Els, "A universal Homophonic coding algorithm based on Arithmetic coding," *in Proc. IEEE South African Symposium on Communications and Signal Processing*, pp. 52-59, 1994.

[5] S.P.Mohanty, "Digital Watermarking: A Tutorial Review," Tampa: Department of Computer Science and Engineering, University of Florida, 1999.

[6] A.Kejariwal, "Watermarking," *IEEE Potentials*, vol. 22, no. 4, pp. 37-40, 2003.

[7] P.Wayner, *Disappearing Cryptography-Information Hiding:Steganography & Watermarking*, 2nd ed San Francisco: Morgan Kaufmann Publishers, 2002, p. 271.

[8] M.A. Suhail and M.S. Obaidat, "Digital watermarking-based DCT and JPEG model," *IEEE Transactions on Instrumentation and Measurement*, vol. 52, no. 5, pp. 1640-1647, 2003.

[9] S. Lee and S. Jung, "A survey of watermarking techniques applied to multimedia," in Proc. *IEEE International Symposium on Industrial Electronics*, vol.1, pp. 272-277, 2001.

[10] S. Katzenbeisser and F.A.P. Petitcolas, *Information hiding techniques for steganography and digital watermarking*, Norwood, MA: Artech House, 2002, pp. 121-132.

[11] J.G. Proakis and M. Salehi, *Communication Systems Engineering*, 2nd ed, New Jersey: Prentice Hall, 2002, pp323-324.

[12] M.S. Hsieh, D.C. Tseng and Y.H. Huang, "Hiding digital watermarks using multiresolution wavelet transform," *IEEE Transactions on Industrial Electronics*, vol. 48, no. 5, pp. 875-882, 2001.

[13] S. Voloshynovskiy et al, "Attacks on Digital Watermarks: Classification, Estimation-Based attacks and Benchmarks," *IEEE Communications,* pp 118-126, 2001.

[14] P.H.W. Wong and O.C. Au, "A capacity estimation technique for JPEG-to-JPEG image watermarking," *IEEE Transactions on Circuits and systems for video technology*, vol. 13, no. 8, pp. 746-752, 2003.

[15] Y. Choi and K. Aizawa, "Digital watermarking using inter-block correlation: extension to JPEG coded domain," in *Proc. IEEE Int. Conf. Information Technology: Coding and Computing*, pp. 133–138, 2000.

[16] W. Luo, G. L. Heileman, and C. E. Pizano, "Fast and robust watermarking of JPEG files," in *Proc. IEEE 5th Southwest Symp. Image Analysis and Interpretation*, pp. 158–162, 2002.

[17] P. H. W. Wong and O. C. Au, "Data hiding and watermarking in JPEG compressed domain by DC coefficient modification," in *Proc. SPIE Security and Watermarking of Multimedia Contents*, vol. 3971, pp. 237–244, 2000.

[18] P. H. W. Wong and O. C. Au, "Data hiding technique in JPEG compressed domain," in *Proc.SPIE Security and Watermarking of Multimedia Contents*, vol. 4314, pp. 309–320, 2001.

[19] G.K Wallace, "The JPEG still picture compression standard," *IEEE Transactions on Consumer Electronics*, 1991.

[20] DCube Software Technologies, "JPEG Baseline compression", http://www.funducode.com/conferences.asp. Last accessed on June 2005.

[21] W.B. Pennebaker and J.L. Mitchell, *The JPEG Still Image Data Compression Standard*, Van Nostrand Reinhold, 1993.

[22] Image Compression-JPEG, http://www.jpeg.org/JPEG. Last accessed on November 2005.

[23] D. A. Huffman, "A Method for the Construction of Minimum Redundancy Codes," *Proceedings of the IRE*, vol. 40, pp. 1098–1101, 1952.

[24] CCITT (ITU Recommendation T.8), *Digital Compression and Coding of Continuous tone Still Images, Part 1, Requirements and Guidelines*, ISO/IEC JTC1 Draft International Standard 10918-1, Nov. 1991.

[25] G.Langdon and J. Rissanen, "Compression of Black-White Images with Arithmetic Coding," *IEEE Transactions on Communications* , vol 29, no 6, pp. 858-867, 1981.

[26] G. G. Langdon, "An Introduction to Arithmetic Coding," *IBM J. Res. Develop*., vol. 28, pp. 135–149,1984.

[27] J. Rissanen and G. Langdon, "Arithmetic coding," *IBM J. Res. Develop*., vol. 23, no. 2, pp. 149-162, 1979.

[28] A. Moffat, R. M. Neal and I. H. Witten, "Arithmetic Coding Revisited," *in Proceedings of the Data Compression Conference*, pp. 202-211, 1995.

[29] P.G Howard and J.S Vitter, "Arithmetic coding for data compression," *in Proceedings of the IEEE* , vol.82, no. 6 , pp. 857-865, June 1994.

[30] D.L Duttweiler and C. Chamzas, "Probability Estimation in Arithmetic and Adaptive-Huffman Entropy Coders," *IEEE Transactions on Image Processing*, vol.4, no.3, pp.237-247, 1995.

[31] N. Abramson, *Information Theory and Coding*, New York: McGraw Hill, 1963, pp. 61-62.

[32] P. Etherington, "Source Coding and Data Compression," Department of Electrical, Electronic & Computer Engineering, Heriot-Watt University, Edinburgh, 2002.

[33] B.Zhu, E.Y. Yang and A.H. Tewfik, "Arithmetic Coding with Dual Symbol Sets and its Performance Analysis," *IEEE transaction on Image Processing, vol.8, no. 12*, pp. 1667-1999, December 1999.

[34] W.T Penzhorn, "Homophonic coding of Cryptographic sources," *in Proceedings of IEEE AFRICON Conference*, *28 Sept.-1 Oct*, vol.1, pp.87-92, 1999.

[35] W.Stallings, *Cryptography and Network Security Principles and Practices*, 3rd ed, New Jersey: Prentice Hall, 2003, pp. 35.

[36] B. Ryabko and A. Fionov, "Efficient Homophonic Coding," IEEE transactions on Information *Theory*, vol. 45, no. 6, pp. 2083-2091, 1999.

[37] H. N. Jendal, Y. J. B. Kuhn, and J. L. Massey, "An Information-Theoretic Treatment of Homophonic Substitution," *Advances in Cryptology - Eurocrypt '89, (LNCS 434), Springer-Verlag*, pp. 382–394, 1990.

[38] C. Shannon, "Communication theory of secrecy systems," *Bell Syst. Tech. J.*, vol. 28, no.4, pp. 656 – 715, 1949.

[39] W.B Pennebaker and J.L. Mitchell, "Probability Estimation for the Q-coder," *IBM J. Res. Develop.*,vol.32, no.6, pp.732-752,1988.

[40] W.B. Pennebaker, J.L. Mitchell, G.G. Langdon, Jr. and R.B.Arps, " An overview of the Basic principles of the Q-coder Adaptive Binary Arithmetic Coder," *IBM J. Res. Develop.*,vol.32, no.6, pp.717-726,1988.

[41] J.L Mitchell and W.B Pennebaker, "Software Implementations of the Q-coder," *IBM J. Res. Develop.*,vol.32, no.6, pp.753-774,1988.

[42] S. Samuel and W.T. Penzhorn, "Digital Watermarking for Copyright Protection," *IEEE Africon Conference, Botswana*, vol. 2, pp. 953-957, 2004.

[43] J.J. Rissanen and G. Langdon, "Universal Modeling and Coding," *IEEE Transactions on Information Theory,* pp.12-23, 1981.

[44] C.G. Gunther, "A universal algorithm for homophonic coding," *Advances in Cryptology Eurocrypt-88 (LNCS 330), Springer-Verlag*, pp. 405-414, 1988.

[45] W.T. Penzhorn, "A fast homophonic coding algorithm based on arithmetic coding," *Fast Software Encryption (LNCS 1008), Springer-Verlag*, pp.329-345, 1995.

[46] V.C. da Rocha,Jr. and J.L Massey, " On the entropy bound for optimum homophonic substitution," *Proceedings of the IEEE Int. Symposium on Information Theory*, pp.93, 1997.

[47] R.L. Milidiu, C.G. de Mello, J.R. Fernandes, "Adding Security to Compresses Information Retrieval Systems," *Computer Department PUC-Rio, Brazil*, 2001.

[48] D. Kruger and W.T. Penzhorn, "A novel method of homophonic coding to increase E-Commerce security," *SATNAC 2001 Conference, South Africa, September,* 2001.

[49] D. Kruger, "Adaptive homophonic coding techniques for enhanced Electronic-Commerce security," *M.Eng (Electrical) Department of Electrical, Electronic and Computer Engineering, University of Pretoria, South Africa*, 2002.

[50] T. C. Bell, J. G. Cleary, and I. H. Witten, *Text Compression,* Prentice Hall, 1990.

[51] C. Boyd, "Enhancing secrecy by data compression: theoretical and practical aspects," *Advances in Cryptology- Eurocrypt, (LNCS 547), Springer-Verlag,* pp. 266-280, 1991.

[52] C.C Stevens, "Homophonic Coding Report," *Department of Electrical and Electronic Engineering, University of Pretoria,* 1992.

[53] Break Point Software, Hex Workshop v3.11, http://www.bpsoft.com. Last accessed on June 2005.

[54] Independent JPEG Group Library, http://www.ijg.org. Last accessed on March 2006.

[55] X. Xue and W. Gao, "Arithmetic coding with improved solution for the carry-over problem,"*Proceedings of Data Compression Conference, 25-27 March1997,* pp.476, 1997.

[56] P.G. Howard and J.S. Vitter, " New methods for lossless image compression using arithmetic coding," *Proceedings of Data Compression Conference, 8-11 April 1991,* pp. 257-266, 1991.

[57] G.I. Sada and B. Usevitch, "A practical binary 'bit stuffing' method for multiplierless arithmetic encoders," *Symposium on Circuits and Systems,* vol.2, pp. 1005-1007, Aug 1999.

[58] M. Nelson, *The Data Compression Book*, M&T Books, Redwood City, CA,1991.

[59] G. Hudson, H.Yasuda and I. Sebesteyn, "The international Standardisation of Still Picture Compression Technique," *Proceedings of the IEEE Global Telecommunications Conference*, pp. 1016-1021, Nov 1988.

[60] DCube Software Technologies, "JPEG File layout and Format", http://www.funducode.com/conferences.asp. Last accessed on June 2006.

[61] W.D. Withers, " A rapid probability estimator and binary arithmetic coder," *IEEE transactions on Information Theory,* vol. 47, no. 4, pp. 1533-1537, 2001.

[62] R. Ansari and N. Menon, " The JPEG Standard," *Department of Computer Science, University of Illinois and Polytechnic University, Chicago & New York*, Feb 1999.

[63] M. van Droogenbroeck and J. Delvaux, " An entropy based technique for information embedding in images," *Department of Electrical Engineering and Computer Science, University of Liege, Belgium*,

[64] Jpegcrop, http://jpegclub.org. Last accessed on August 2005.

[65] G. Vollbeding, JPEG arithmetic encoding and decoding portable software implementation, http://sylvana.net/jpeg-ari/jpeg-ari.zip. Last accessed on July 2005.

[66] V. Ratnakar and O.G. Guleryuz, "Standards compliant watermarking for access management," *Epson Palo Alto Laboratory, USA*, 2001.

[67] R.B. Wolfgang, C.I. Podilchuk and E.J. Delp, "Perceptual Watermarks for Digital Images and video," *Proceedings of the IEEE,* vol. 87, no.7, pp. 1108-1126, Jul 1999.

[68] N.S. Jayant, J.D. Johnson and R.J. Safranek, "Signal compression based on models of human perception," *Proceedings of IEEE*, vol. 87, pp. 1385-1422, Oct 1993.

[69] S. Burgett, E.Koch and J.Zhao, "Copyright labelling of digitized image data," *IEEE Communications Mag.,* vol.36, pp. 94-100, Mar 1998

[70] W.Bender, D.Gruhl, N. Morimoto and A.Lu, "Techniques for data hiding," *IBM Syst.J,*. vol. 35, nos. 3&4, pp. 313-336, 1996.

[71] S.Craver, N. Memon, B.L. Yeo and M.M. Yeung, "Resolving rightful Ownerships with invisible watermarking techniques: Limitations, Attacks and Implications," *IEEE Journal on Selected Areas in Communication,* vol. 16, no. 4, pp. 273-286, May 1998.

[72] F.M. Boland, J.J.K. O' Ruandaidh and C.Dautzenberg, "Watermarking digital images for copyright protection," *Proceedings of IEEE Image Processing Applications Conference,* pp. 326-330, 1995.