Image Steganographic System using Wavelet/Contourlet Transforms and Blowfish Encryption

by

Saddaf Rubab
2009-NUST-MSPhD- CSE (E)-10
MS-09 (SE)

Submitted to the Department of Computer Engineering in fulfillment of the requirement for the degree of

Masters of Science
In
Computer Software Engineering

Thesis Supervisor
Prof. Dr. Muhammad Younus Javed

College of Electrical and Mechanical Engineering
National University of Science and Technology
2012

Acknowledgments

All praise to Allah, the most beneficent and most merciful.

I would like to express my gratitude to my supervisor, Brig Dr. Muhammad Younus Javed, whose expertise, understanding, and patience, added considerably to my graduate experience. I appreciate his vast knowledge. I would like to thank the other members of my committee, Lt. Col. Dr. Almas Anjum, Dr. Assia Khanum, and Dr. Saad Rehman for the assistance they provided at all levels of the research project.

I would also like to thank my parents and husband for constantly encouraging me to complete my thesis, especially my husband for providing me a facilitating environment for completing my research. I would not have been able to complete this thesis without the support of these people.

Dedicated to my loving parents and husband…

Abstract

Steganography is a means to hide the existence of information exchange. Using this technique the sender embeds the secret information in some other media. This is done by replacing useless data in ordinary computer files with some other secret information. The secret information could be simple text, encoded text or images. The media used as the embedding plane could be an image, audio, video or text files. Using steganography ensures that no one apart from the sender and the receiver knows about the existence of the message.

In this thesis, a steganography method based on transforms used i.e. Wavelet and Contourlet. Devised algorithm was used against each transform. Blowfish Encryption method is also embedded to double the security impact.

The major advantage of applying transforms is that the image quality is not degraded even if the number of embedded characters is increased. The proposed system operates well in most of the test cases. The average payload capacity is also considerably high.

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF TABLES

Chapter 1

Introduction to Information Hiding

1.1 Introduction

Steganography is an art of hiding information in some other media of information. The secret information is embedded into another media in such a way that no one else than sender and propspective reciever can know about the presence of the secret information. In other words, using this technique even obscure that a secret information is being transmitted [1].

Merely this fact makes Steganography an essential part of modern day digital communications. The other technique that is used for secret communication is 'Cryptography' in which the sender uses an encryption key to hide the message, this hidden message is transmitted using any public channel, and the decode the original message is possible only if the receiver has the correct decryption key [3].

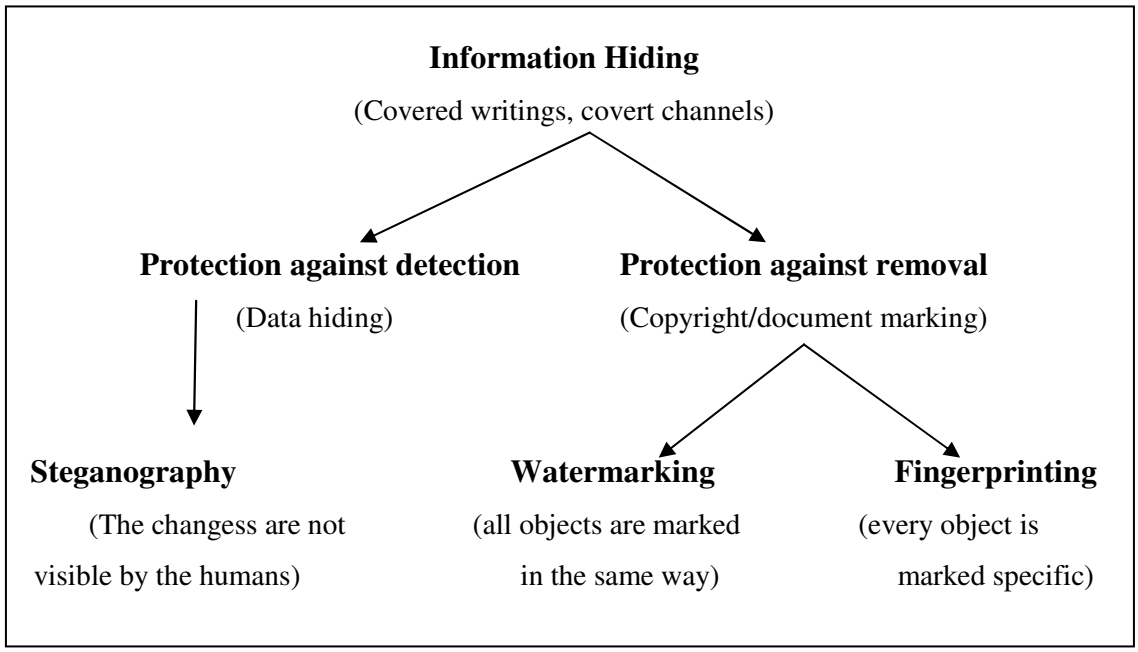


Figure 1.1: Information Hiding

Following are the directions of information hiding:-

a. **Protection against detection**

   i. This is achieved using schemes that do not transform the original unmarked object in a perceptible manner; the changess are not visible to human eye or computers.

b. **Protection against removal**

   These techniques emphasize that schemes should be robust to attacks by enemies; however to remove the hidden message is impossible without degrading the object's quality. They are two different techniques for protection against removal. In both cases, the information should be invisible to anyone, but it should be very difficult to remove it.

   i. Watermarking is the process of embedding marks in any sounds, images, etc for l ater identification e.g., marking a bank note.

   ii. Fingerprinting is the process of embedding a serial number into every copy of an object. e.g., detection of breaks in license agreements.

If we want to achieve the confidentiality in any application, we have two options, either to opt encryption or steganography. If encryption is used, only the receiver can read the secret message, but any third party can get aware of secret communication being carried out. If steganography is used, the presence of any secret message is hidden.

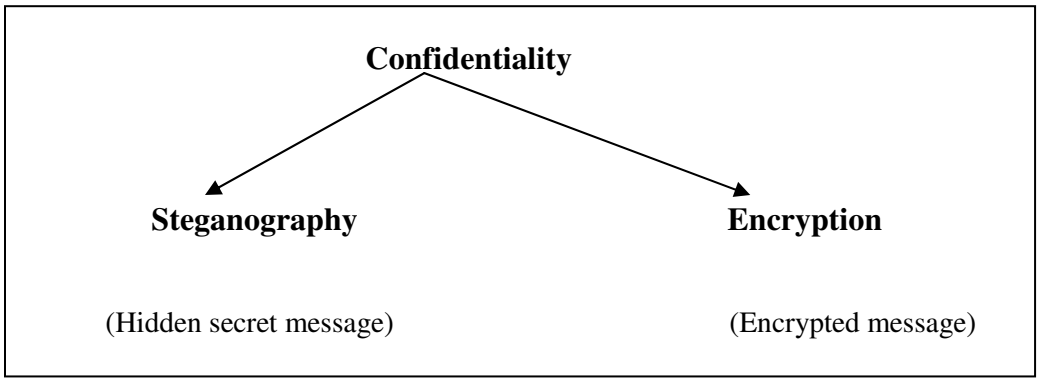


Figure 1.2: Achieving confidentiality

To protecting contents of object is another main issue. The sender wants to hide the secret message or to make it difficult for anyone to get it. Hiding the messages is done by using digital signatures. Unlike digital signatures, the watermarks used for hiding messages are spread over the covered text, and keeping the length of message remains same.

1.2 History of Steganography

Many methods and techniques have been devised and used to hide the messages. Secret Warfare; The Battle of Codes and Ciphers discusses several methods of cryptography and stegnaography by Bruce Norman [4].

The “Histories of Herodotus” is one of the most primitive documents with the mention of steganography. Wax covered tables were used to wrte some text in ancient Greece. In order to protect secret messages from enemy attacks one other method used was to ragged the wax from wax tables and write secret messages on underlying wood and tables were then covered again by wax.

Another creative scheme was to shave the head of the messenger. The blad was then used to print a tattoo either in a form of message or image on the shaved head. The messenger’s hair was then allowed to grow so that the tattoo could be covered. This used to make the message undetectable unless the messenger’s head was shaved again.

Invisible inks were also in practice. Like in World War II invisible inks played a great role for hidden communications. A letter may contain secret messages written in between the lines of letters. To get the hidden messages the ink was heated it turned dark and shows the messages to reciever. These were produced by vinegar, milk or fruit juices.

As the time advanced, new sophisticated invisible inks were developed that react to various chemicals. Unecrypted messages like null chiphers were used in which the message to be send secretly is camouflaged in a simple text.

By advancements in detecting secret messages, need also emerged to develop new technologies to pass more amount of information/message and at the same time enemies or third parties remain unaware. Microdot technology was also in practice. Basically, the microdots are pictures equal to the size of printed typewritten pages. In this the message is not an encrypted one. It is small enough to not draw attention of the reader.

Steganography has many real time applications like steganographic softwares are welcomed for its data hiding beauty. In computers the images are replresented as arrays of numbers. This representation can easily be exploited by simply replacing the Least Significant Bits (LSB) of image (arrays of numbers)

with the secret message bits. LSB is an easy approach but one should keep in mind that human eye shall not detect that image is altered.

It is so far observed that hiding data dates back to World War II. Since then many methods and techniques evoloved including writing on wood underlying the wax to the secret key methods.

1.3 Techniques

Figure 1.3 gives adetailed look at information hiding techniques.

1.3.1 Steganography

It is an art of communication in such a way that presence of information is not detectable[1]. The word steganography comes from Greek meaning (covered writing). It is an art of encoding secret messages in a media while keeping media imperceptible to its audience.

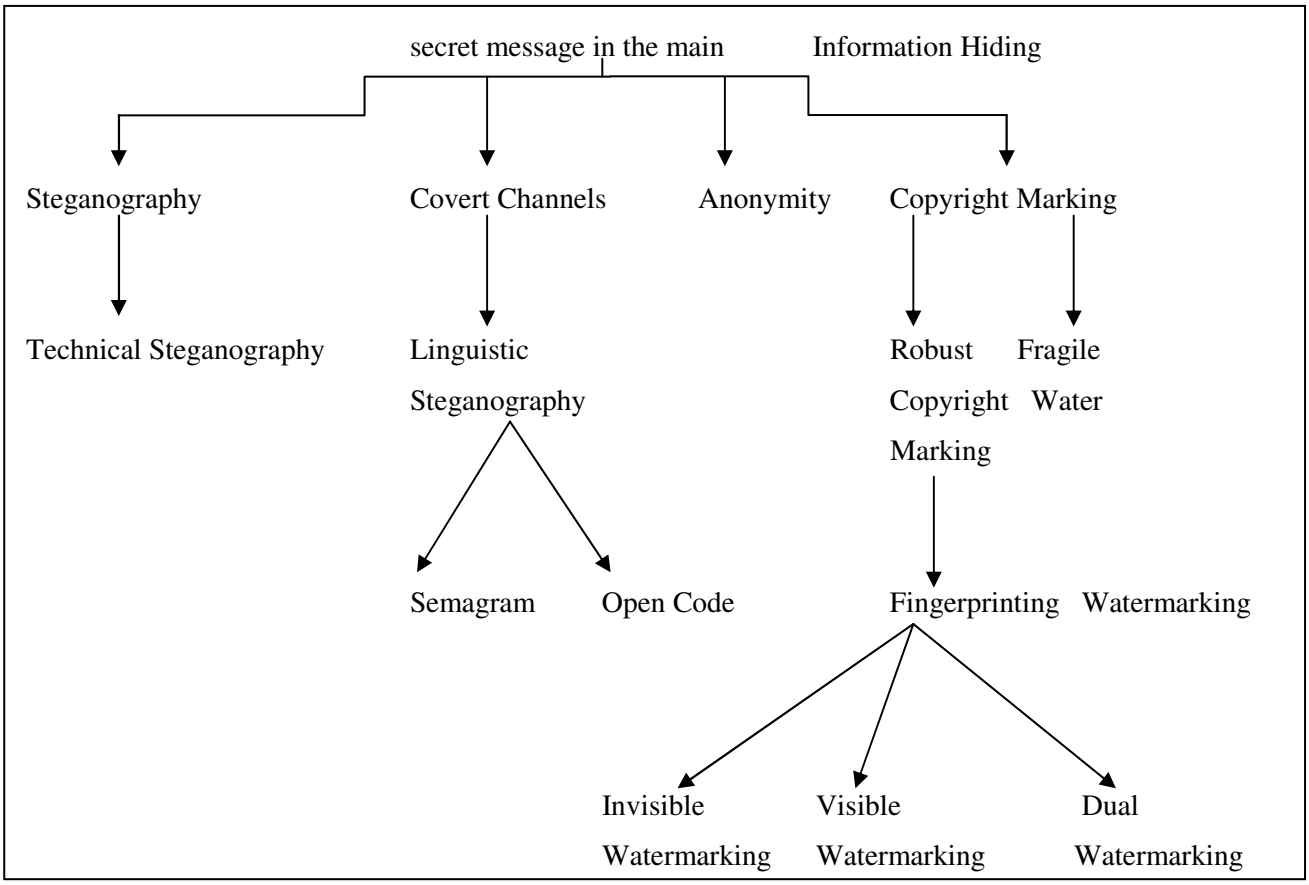


Figure 1.3: Information Hiding Techniques

This way it also prevents secret messages from attacks by third party. It establish covert channels for communication. These covert channels are key paths for transmitting secret message.

1.3.2 Watermarking

It is a technique to embed object which inturns known as watermark or label in any digital object. The name "watermark" is derived from the slightly visible marks embossed on organizational stationery.It works on principle that watermark can be detectable to make a statement about the embedded object. The digital object can be any image, video, text or audio.

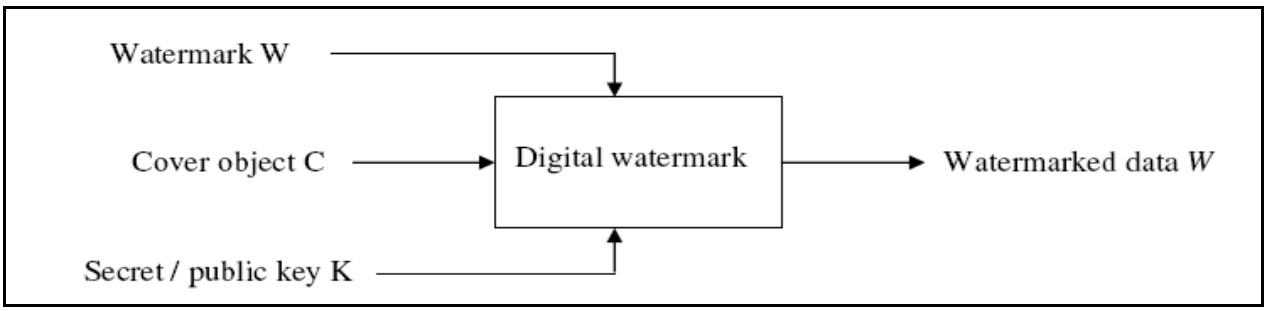


Figure 1.4: Digital watermarking – Embedding

The watermark is a recognizable object which is used to provide protection of a cover digital media. We have various types of watermarks like visible or invisible. These two have their own benefits. Examples are logos of companies, universities and sometimes only two tone dots to explain the copyright information.

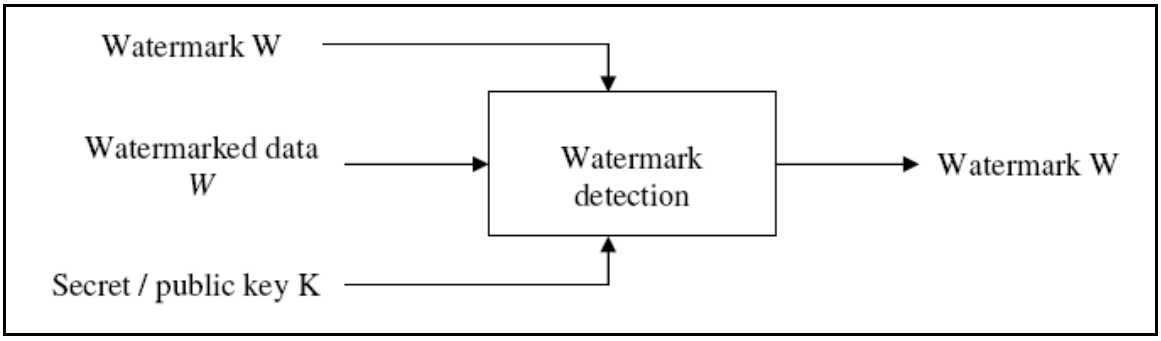


Figure 1.5: Digital watermarking – Decoding

1.3.3 Fingerprinting & Labelling

Fingerprints are sometimes called as labels. Digital watermarking differs from "digital fingerprinting". Fingerprinting is the process of adding fingerprints to an object and recording them, or identifying and

recording fingerprints that are already intrinsic to the object making that object unique as compared to other similar objects. Digital fingerprinting produces a metafile that describe the contents of the source file. [5]

1.3.4 Digital Signature

A digital signature is based upon the idea of asymmetric cryptography [6]. A secret key is used to encrypt a hashed version of an image to form a signature for the used image and only the sender knows the secret key. At reciever end a coupled secret key is used to decrypt the formed signature. The image is first hashed using the hashing function used for ecryption. If the formed hashes match, the image is authetic. It is used in proofing the first authorship in combination with timestamp.

1.3.5 Cryptography

Cryptography is an art that consists of mangling information into apparent unintelligible information. It works on he principle of sending secret information in between sender and reciever in such a way that any intermediate party is unaware of the information. The intermediate or third party is ware of the secret communication being carried out but still not able to know about the secret messages.

1.3.6 Encryption

In a process to camouflaged a message to hide its presence is called encryption. It requires an encoding scheme and ecryption key which may or may not be same for encoding and decoding. In this way sender and reciever parties can use a same key to send secret messages in between and having secure communication. Any third party want to intercept, needs a secret key to decode which was used to decode the message. Attacks on encrypted data can be like modifying, analyzing, false/fake replying or cutting down.

1.4 Steganography versus Cryptography

The term steganography means “covert writing” whereas cryptography means “secret writing”. In cryptography only recipent can remove the camouflaged message and read the secret information. In encryption the message is protected. In steganography the message is hidden in any media and it can be transmitted in an open channel . It donot require any covert channel. Any other media plays the role of carrier for secret message.

Cryptography techniques “scramble” the original message so if intercepted, the messages can not be understood by any unintended recipient. steganography, in an essence, “camouflages” a message to hide its presence and makes it seem “invisible” thus concealing the fact that a message is being sent. An encrypted message may draw suspicion while an invisible message does not.

Steganography is not intended to replace cryptography rather it supplements it. The two techniques could be combined to provide a two tier protection to the secret communication. Hiding an encrypted message with a Steganographic method ensures that if the Steganographic message is detected then the interceptor also needs to decrypt that message.

1.5 Steganography versus Digital Watermarking

Basically the purpose of use differs the two techniques. A watermark acts like an attribute of the cover media. It contains information about copyrights, authorship etc. But in steganography the secret message embeded has no relation to cover media. We consider robustness of steganography as main feature than in watermarking [2].

Steganography is the act of embedding a secret message to any cover media. It is like encrypting a message, but instead of running it through a cipher, the message is broken up and stored in unused, or unnoticeable, bits within the overall image.

1.6 Categories of Steganography

Steganography algorithms can be classified in several categories of increasing quality [15] [16]:-

a. Adding data at the end of the carrier file.
b. Inserting data in some junk or comment field in the header of the file structure.
c. Embedding data in the carrier byte stream, in a linear, sequential and fixed way.
d. Embedding date in the carrier bytes stream, in a pseudo-random way depending on a password.
e. Embedding data in the carrier byte stream, in a pseudo-random way depending on a password, and changing other bits of the carrier file to compensate for the modifications induced by the hidden data, to avoid modifying statistical properties of the carrier file.

1.7 Need of Digital Steganography

The major purpose of steganography is to hide the secret data that is normally encrypted to avoid interception. However, the main problem with encryption is that it transforms intelligible information to un-structured random data which is quite rare in today's computer world. From small TCP/IP packet to largest files on hard disks everything is stored and transmitted in strict hierarchical structure. So any random data suddenly appearing within a stream of structured data is likely to draw attention. The objective of using steganography is just opposite.

1.8 Structure of Steganographic System

Steganography embeds a secret message in a cover object, this process is usually accompnied by a secret key, and reciever needs this key to intercept the message. Figure 1.6 gives a generic representation of steganographic system.

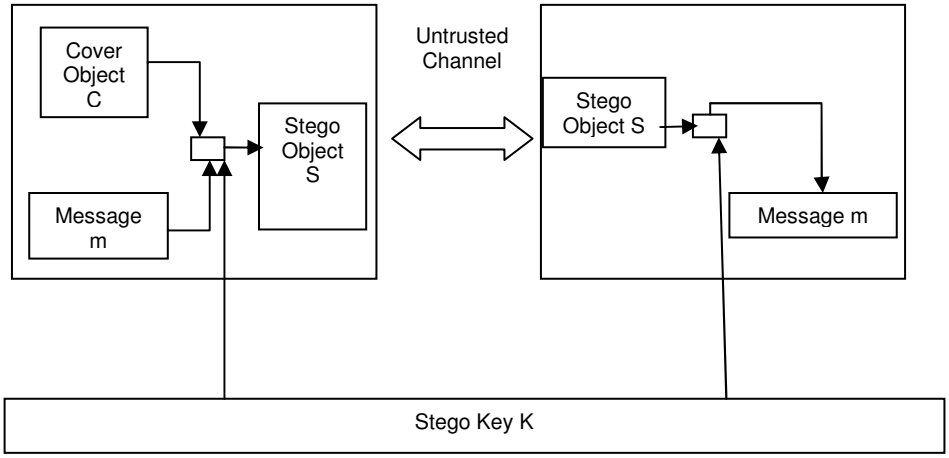


Figure 1.6: Steganographic System

1.9 Requirements of Steganography

For the steganographic system we have many requirements which need to be fulfiled. The following is the list of main requirements that steganography technique must satisfy.

1.9.1 Capacity

When there is a large size of secret information, capacity is the main factor to consider. For example, medical images, the data of patients and treatments suggested could be embedded in a single image.

1.9.2 Imperceptibility

Imperceptibility is defined as not getting aware of the presence of secret message. It is important when sender and reciever wants a secret communication in between and it is not detectable by any enemy.

1.9.3 Robustness

Steganographic methods needs secure communication and for this robustness is another main factor. It is defined as the technique's tendency that how much and how it can resist to the attacks by third party or enemies.

1.10 Limitations of Steganography

A major limitation of steganography is the size of covused to hide the secret message. To have successful and secure steganography, the secret message is embeded with less number of alterations to the cover media. This concludes in less capacity for any secret information to be embedded in.

It has a same assumption as for encryption. If Bob and Alice wants a secret communication being taking place then they both should first agree on a method. If using encryption for this case, Bob as reciever know that he will receive a cipher or encrypted text. If using steganographic system, Bob will not be knowing that an image is not steganed or not embedded woth the secret message. Consider a scenario, Alice borrows the digital camera from Bob and somehow or the other didn't tell Bob to read the 73rd byte in images she will send to him. As Bob is totally unaware of this steganographic system, the large numbers of pictures sent will decrease the chance that Bob will allow Alice to borrow his digital camera.

The size of cover media decides how much amount of secret information can be embedded. The fewer constraints that exist on the integrity of the medium, the more potential it has for hiding data. For an instance, we have a simple English language literature of one paragraph only, then it will be a great difficulty to embed a large secret message due to the less number of characters.

1.11 Detecting Hidden Code

Steganalysis, is the known method to undo the steganographic technique applied. It is used for decoding the hidden message from a given cover media. Two major tools in Steganalysis, information theory and statistical analysis, reveal in clear terms the tremendous potential for hidden information in Internet data — as long as a set of data can be compressed to a smaller size, there is room for hidden data within the

medium. Accordingly, the path of seeking hidden data is untrustworthy and vague. Unless hidden data is encoded in a common, well-defined format, it may be virtually impossible to detect in the carrier data. In other words, a set bit can represent the any specific information, depending on how I choose to define my encoding. A Steganalysis expert is unable to determine the chosen encoding, for him a bit is just a bit.

Steganalysis, has not received the attention it deserves. The ability to control sensitive information is a critical part of maintaining a large institution. Steganalysis research has been stimulated by the increasing number of tools available for digital steganography. Steganalysis remains difficult to perform with great accuracy on some media. The growing field of cyber forensics — detective works in the digital domain —should create greater demand for Steganalysis tools in the near future.

Chapter 2
Steganographic Approaches and Encryption Algorithms

2.1 Introduction

In a digital world, steganography and cryptography both are used for providing security to secret information so that it could only be shared with the intended recipients. Both of these techniques could be used but none of them is completely secure. However, as most of the experts would suggest, it is better to use both in conjunction in order to provide multiple layers of security to the secret communication.

History of steganography dates back to 440BC where a slave was used as a carrier [32], but in modern world, it is usually implemented computationally, where text, audio, image or protocol served as cover medium. I have limited the scope to image steganography as there has been a great interest in image as cover media. Image steganography have been implemented and discovered in the categories of spatial and transform domain.

Multiple data formats are in use for steganography these days such as .bmp, .doc, .gif, .jpeg, .mp3, .txt and .wav. These data formats are used for steganography because they are widely used on the internet. These data formats provide the capability of embedding secret data by replacing the noise or redundant data. Keeping in view its use, steganography promises a bright future in the world of digital communications and privacy on open systems such as the Internet. Steganographic research aims at removing the weaknesses of the cryptographic systems in order to ensure complete security. Steganography can play an important role by hiding the secret message within other files. This increases the security to a large extent as only the parties having secret communication can know about the existence of the secret message. To make the systems fool proof, multiple layers of security could be added by using cryptography and steganography together.

In this chapter different types of steganography is covered along with some of the other principles that are used in steganography.

2.2 Steganographic Approaches

Let's first try to explain steganography with the help of an example. Let's consider three friends named A, B and C. A wants to send a secret message (M) to B using a random (R) cover message to create a cover (C) which can be sent to B without raising suspicion. A then changes the cover message (C) to a stego-object (S) by embedding the secret message (M) into the cover message (C) by using a stego-key (K). A then sends the stego-object (S) to B and C will not be able to detect the secret message. B could then read the hidden message (M) because he already knows the stego-key (K) used to embed the message into the cover message (C).

As Fabien A.P. Petit colas [15] points out, in a 'perfect' system, a normal cover should not be discernible from a stego-object, neither by a human nor by a computer if various statistical patterns are considered. However, practically this is not always the case. A secret message could only be embedded into the cover image if it contains considerable amount of noise or redundant data. This is because the steganography actually replaces noise with the secret message. This limits the types of data that can be used with steganography.

2.2.1 Types of Steganographic Protocols

There are three types of Steganographic protocols in use:-

a. Pure Steganography
b. Secret Key Steganography
c. Public Key Steganography

2.2.1.1 Pure Steganography

This type of steganography is defined as a system that does not require the exchange of a cipher such as a stego-key. This technique is not as much secure as compared to rest of the techniques. Because in this case, the sender and receiver have to rely upon the presumption that no one else is aware of this secret message.

2.2.1.2 Secret Key Steganography

It is defined as a steganographic system that requires the exchange of a secret key (stego-key) prior to communication. This technique takes the cover object and embeds the secret message by using a secret key (stego-key). Using this methodology ensures that only the parties who know the secret key can read the secret message. In contrast to pure steganography where a perceived imperceptible communication channel is present, secret key steganography exchanges a stego-key, which makes it more prone to

interception. But it is still secure in a sense that only the parties who know the secret key can extract the secret message.

2.2.1.3 Public Key Steganography

Public key steganography is defined as a steganographic system that uses a public and a private key pair to protect the secret communication between the two parties. The sender will use the public key during the embedding process and only the private key, which is mathematically related to the public key, can decrypt the secret message. This method of steganography provides a more robust way of implementing a Steganographic system because it has multiple levels of security. As a result the interceptor must first suspect the use of steganography and then he would need a way to break the algorithm used by the public key system before they could actually read the hidden message.

2.2.2 Techniques in Use

The major techniques used for embedding secret information are discussed below:-

2.2.2.1 Embedding Secret Messages in Text

Embedding secret messages in text files is a very difficult task. This is because text files don't have enough amount of redundant data to be substituted with secret message. Another drawback is that any third party can alter the hidden message by changing the text itself or reformatting the text to some other form (from .TXT to .PDF, etc.). There are a variety of methods available to achieve text based steganography. A few of the methods are briefed below:-

2.2.2.1.1 Line-shift encoding

As the name suggests, comprises of shifting each line of text vertically up or down by as little as 3 centimeters. A secret message could be embedded depending on whether the line was up or down from the stationary line [16].

In this method, the lines of the text are vertically shifted to some degree (for example, each line shifts (1/300 inches up or down) and information is hidden by creating a unique shape of the text. This method is proper for printed texts. However, in this method, the distances can be observed by using special instruments of distance assessment and necessary changes can be introduced to destroy the hidden information. Also if the text is retyped or if character recognition programs (OCR) are used, the hidden information would get destroyed.

2.2.2.1.2 Word-shift encoding

This method of encoding is more secure as it is less visible than line-shift encoding. However, an important requirement of this method is that the text being used supports variable spacing [15].

In this method, by shifting words horizontally and by changing distance between words, information is hidden within the text. This method is acceptable for texts where the distance between words is varying. This method can be identified less, because change of distance between words to fill a line is quite common. But if somebody was aware of the algorithm of distances, he can compare the present text with the algorithm and extract the hidden information by using the difference. The text image can be also closely studied to identify the changed distances. Although this method is very time consuming, there is a high probability of finding information hidden in the text.

2.2.2.1.3 Feature specific encoding

In this method, some of the features of the text are altered. For example, the end part of some characters such as h, d, b or so on, are elongated or shortened a little thereby hiding information in the text [13]. In this method, a large volume of information can be hidden in the text without making the reader aware of the existence of such information in the text. This is up till now the most secure encoding method. It is difficult to intercept as each type of formatted text has quite a lot of features that can be used for embedding the secret message. By placing characters in a fixed shape, the information is lost. Retyping the text or using OCR program (as in methods 1 and 2) destroys the hidden information.

2.2.2.1.4 Semantic Methods

In this method, synonyms for certain words are used thereby hiding information in the text [14]. A major advantage of this method is the protection of information in case of retyping or using OCR programs. However, this method may alter the meaning of the text.

2.2.2.1.5 Abbreviations

Another method for hiding information is the use of abbreviations. In this method, very little information can be hidden in the text. For example, only a few bits can be hidden in a file of several kilobytes [11].

2.2.2.1.6 Open Spaces

In this method, hiding information is done through adding extra white-spaces in the text. These whitespaces can be placed at the end of each line, at the end of each paragraph or between the words [12].

This method can be implemented on any arbitrary text and does not raise attention of the reader. However, the volume of information hidden under this method is very little. Also, some text editor programs automatically delete extra white-spaces and thus destroy the hidden information.

2.2.2.1.7 SMS-Texting

This method is based on the use of abbreviations for steganography in SMS messages [22]. This method can be used on devices such as Pocket PC and PDAs. Also this method can be implemented on desktop PCs using SMS gateway for sending and receiving SMS messages. This technique of text steganography can be developed and privatized in other areas as well. For example, the abbreviated equivalents of engineering terminology can be used in engineering texts.

In addition to English, SMS can be sent in other languages such as Arabic, Greek and so on. Therefore, the abbreviation text steganography method in these languages can also be used for hiding information in SMS. A small amount of information can be hidden in this method. However, the size of input data can be decreased by compression before hiding the data. Cryptography of the intended data can also add the security of this method. Both of these operations need further computations which can not therefore be implemented on any type of mobile phone and if used, the number of mobile phones capable of executing this program will be decreased.

2.2.2.2 Encoding Secret Messages in Audio

Encoding secret messages in audio is the most challenging technique to use when dealing with steganography. This is because the Human Auditory System (HAS) has such a dynamic range that it can listen over. To put this in perspective, the HAS perceives over a range of power greater than one million to one and a range of frequencies greater than one thousand to one making it extremely hard to add or remove data from the original data structure. The only weakness in the HAS comes at trying to differentiate sounds (loud sounds drown out quiet sounds) and this is what must be exploited to encode secret messages in audio without being detected.

There are two concepts to consider before choosing an encoding technique for audio. They are the digital format of the audio and the transmission medium of the audio.

2.2.2.2.1 Digital Audio Formats

There are three main digital audio formats namely Sample Quantization, Temporal Sampling Rate and Perceptual Sampling.

Sample Quantization

It is a 16-bit linear sampling architecture used by popular audio formats such as (.WAV and. AIFF).

Temporal Sampling

This method uses selectable frequencies (in the KHz) to sample the audio. Generally, the higher the sampling rate is, the higher the usable data space gets.

Perceptual Sampling

The last audio format is Perceptual Sampling. This format changes the statistics of the audio hugely by encoding only the parts the listener perceives, thus maintaining the sound but changing the signal. This format is used by the most popular digital audio on the Internet today in ISO MPEG (MP3).

2.2.2.2.2 Transmission medium

Path the audio takes from sender to receiver must also be considered when encoding secret messages in audio. W. Bender [15] introduces four possible transmission mediums:-

a) **Digital end to end** - from machine to machine without modification.

b**) Increased/decreased resampling** - the sample rate is modified but remains digital.

c) **Analog and resampled** - signal is changed to analog and resampled at a different rate.

d) **Over the air** - signal is transmitted into radio frequencies and resampled from a microphone.

2.2.2.2.3 Popular encoding methods

The most popular encoding methods for hiding data inside of audio are low-bit encoding, phase-coding and spread spectrum.

Low-bit encoding

Low-bit encoding embeds secret data into the least significant bit (LSB) of the audio file. The channel capacity is 1KB per second per kilohertz (44 kbps for a 44 KHz sampled sequence). This method is easy to incorporate but is very susceptible to data loss due to channel noise and resampling.

Phase coding

This technique substitutes the phase of an initial audio segment with a reference phase that represents the hidden data. This can be thought of, as sort of an encryption for the audio signal by using what is known as Discrete Fourier Transform (DFT), which is nothing more than a transformation algorithm for the audio signal.

Spread spectrum

This method encodes the audio over almost the entire frequency spectrum. It then transmits the audio over different frequencies which will vary depending on what spread spectrum method is used. Direct Sequence Spread Spectrum (DSSS) is one such method that spreads the signal by multiplying the source signal by some pseudo random sequence.known as a (CHIP). The sampling rate is then used as the chip rate for the audio signal communication.

Spread spectrum encoding techniques are the most secure means by which to send hidden messages in audio, but it can introduce random noise to the audio thus creating the chance of data loss.

2.2.2.3 Embedding Secret Messages in Images

Embedding secret messages in digital images is the most extensive form of steganography in practice these days. The main reason is that this technique utilizes the limited power of human visual system. This method has the capability of embedding any type of data that could be converted into a bit stream. This could include cipher text, image and text image. With the research being put into image based steganography, this field is growing at a very fast pace. Before describing different techniques, a brief explanation of digital image architecture and digital image compression techniques is discussed.

As Duncan Sellers [9] explains; "To a computer, an image is an array of numbers that represent light intensities at various points, or pixels. These pixels make up the images raster data." When dealing with digital images 8-bit and 24-bit per pixel image files are typical. Both are discussed below 8-bit images are relatively small in size. The only short coming of these image files is that they can only have 256 possible colors. This can create problems during the embedding process. For 8-bit images normally gray scale color palette is used such as (.GIF) because its slow change in gradient would be hardly susceptible after the secret message has been embedded. 24-bit images are better suited for steganography. 24 bit images can have a very wide range (over 16 million) colors which makes the secret message hard to detect after the secret message has been embedded. The other advantage is that these 24 bit images have a larger

capacity to embed a secret message. The one major disadvantage is their large size (usually in MB), which makes these images more suspicious than 8-bit digital images (usually in KB) when transmitted over an open system such as the Internet.

2.2.2.3.1 Digital Image Compression

Compression can provide a better alternative for large images such as 24-bit images. There are two types of compression techniques in use namely lossy and lossless.

a. Lossy Compression

Lossy compression such as (.JPEG) compresses the size of a digital image by removing redundant image data and creates a very close approximation to the image. This form of compression is normally used with large digital images as mentioned earlier. It also has some disadvantages as it increases the probability that the secret message will lose parts because lossy compression tends to remove any excess data.

b. Lossless Compression

These techniques, as the name suggests, does not remove any excess data from the image thus keeps the original digital image intact. It is for this reason that it is widely used in steganography. Examples of lossless compression techniques are (.GIF and .BMP). The only disadvantage of this method is that it does not compress the size of the image well.

2.2.2.3.2 Masking and Filtering Techniques

These can also be used for steganography such a watermarking. (i.e. Integrating a company's logo on there content). These methods are mostly used with lossy compression techniques such as (.JPEG). This technique actually extends an image data by masking the secret data over the original data as opposed to hiding information inside of the data. The major advantage of these techniques is that they are impervious to image manipulation.

2.2.2.3.3 Major Categories of Image Steganography

The major categories include :-

a. Least Significant Bit Steganography
b. Spread Spectrum Steganography
c. Steganography using Fractals
d. Transform Domain Steganography

2.2.2.3.3.1 Least Significant Bit (LSB) Encoding

This is by far the most popular form of coding technique used for digital images. In this method last bit of every byte is altered to embed one bit of the secret message. Using this technique, 3 bits can be stored in a single pixel from a 24-bit image and one bit in each pixel of an 8-bit image. This is why a24-bit image has greater capacity to hide a message as compared to an 8-bit image. Similarly if a gray scale image is used then two least significant bits can be used to store bits of the secret message [10]. The changes made to the cover image will not be detectable by the human visual system. The only drawback of this system is that it vulnerable to changes in image formats like changing from GIF to JPEG. To hide information in the LSBs of each byte of a 24-bit image, it is possible to store 3 bits in each pixel.

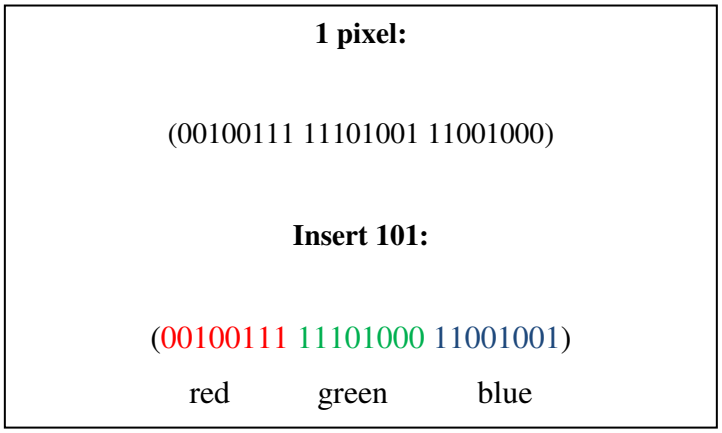


Figure 2.1: A simplified example with a 24-bit image

LSB insertion works well with gray-scale images as well. It is possible to hide data in the least and second least significant bits and the human eye would still not be able to discern it.

Unfortunately LSB insertion is vulnerable to slight image manipulation such as cropping and compression. For example, converting a GIF or a BMP image, which reconstructs the original message exactly (lossless compression), to a JPEG format, which does not (lossy compression), and then converting back, can destroy the data in the LSBs. The best known technique is modifying the Least Significant Bit (LSB) of images. This technique changes the LSBs in an image, as if there is some noise added but the original image is not altered. There are some very efficient ways to do this that even effect of noise is also not visible by human eye.

In practice LSB technique exchanges the LSB of images with LSB of messages that are to be encoded. Fridrich et al. [46] proposed an approach for embedding in spatial domain. In their method, noise that statistically resembles common processing distortion, e.g., scanner noise, or digital camera noise, is

introduced to pixels on a random walk. The noise is produced by a pseudo random noise generator using a shared key. A *parity function* is designed to embed and detect the message signal modulated by the generated noise.

Nameer N. Emam [47] in 2007 proposed an algorihm using LSB insertion in a way to get highly secure steganography system. In this high security layers have been proposed to make it difficult to break through the encryption of the input data and confuse steganalysis too.

S. K. Bandyopadhyay, Debnath Bhattacharyya, Swarnendu Mukherjee, Debashis Ganguly, Poulami Das [48] in 2008 proposed an approach to hide huge amount of data using LSB steganography technique. In their method, they have first encoded the data and afterwards the encoded data is hidden behind a cover image by modifying the least significant bits of each pixel of the cover image. The resultant stego-image was distortion less. Also, they have given much emphasis on space complexity of the data hiding technique.

Adel Almohammad and Gheorghita Ghinea [49] in 2010 compared the gray and colored images when used as cover images using JSteg and JMQT as test methods and suggested that using color images will increase the embedding capacity.

Juan José Roque, Jesús María Minguet, [55] proposed a new method Selected Least Significant Bit, and improves the performance by selecting the color to use for hiding information in only one of the three colors of cover image using Sample Paris analysis. It selects some LSBs of one pixel color component in image and then changes them with message bits for data hiding purpose. The flow chart of the above scheme is plotted in Figure 2.2

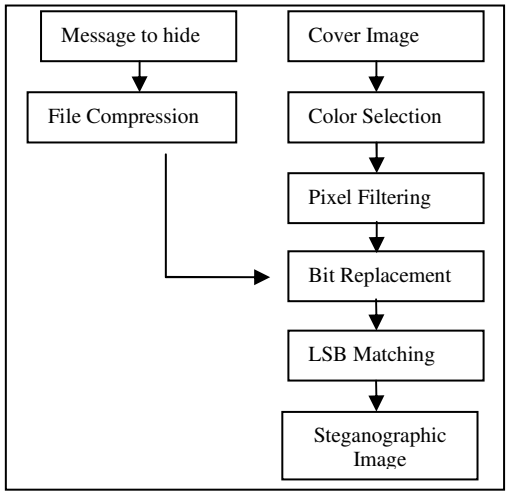


Figure 2.2 Block diagram of [55]

Mrs. Gyankamal J. Chhajed Mrs. Vandana Inamdar Mrs Vahida Attar [50] in 2008 proposed an embedding method in black and white pictures by using blocks of cover images. It uses the flappable blocks to improve the distortion in images and hiding capacity. The cover image is divided in 3x3 blocks. Then possibility of encoding in each block is checked using a lookup table, if its result is positive one bit of message is encoded in that block by changing only one cell. This algorithm performed on basis of picture size to determine the size of flappable blocks and message size to be encoded.

2.2.2.3.3.2 Spread Spectrum Steganography

This system hides and recovers messages of substantial length within digital imagery while maintaining the original image size and dynamic range. The hidden messages can be recovered using appropriate keys without any knowledge of the original image. Image processing, error control coding, and spread-spectrum techniques utilized are described, and the performance of the technique is illustrated. A message embedded by this method can be in the form of text, imagery, or any other digital signal.

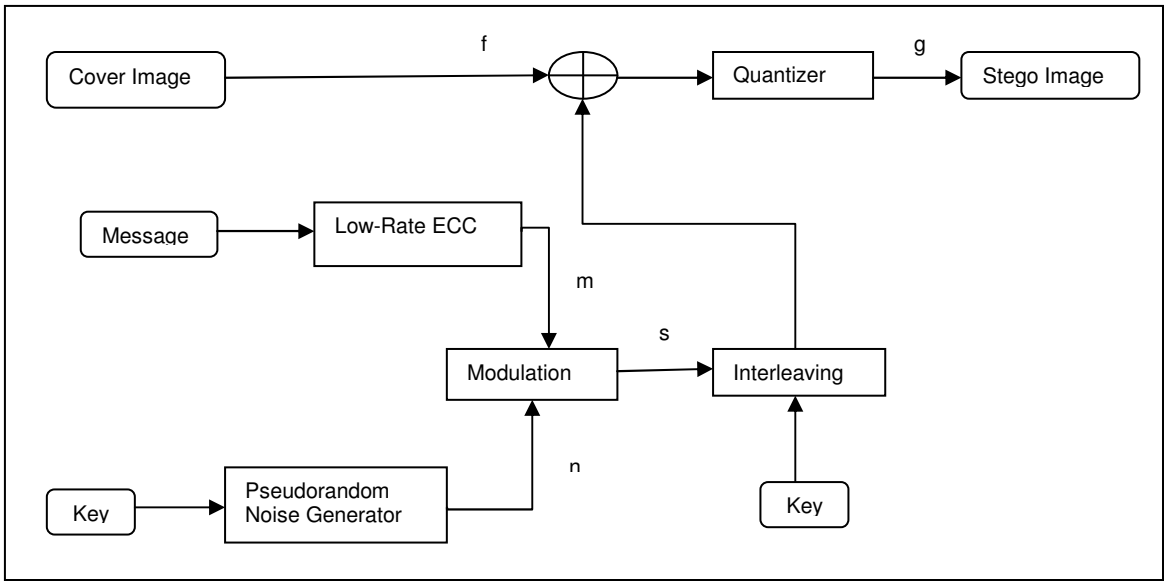


Figure 2.3: Spread Spectrum Encoder

Techniques of spread spectrum communication, error-control coding, and image processing are combined to accomplish SSIS. The SSIS encoder and decoder are shown in Figures 2.3 and 2.4. The fundamental concept of SSIS is the embedding of the hidden information within noise, which is then added to the digital image [3]. This noise is typical of that inherent to the image acquisition process and, if kept at low levels, is not perceptible to the human eye nor is it susceptible to detection by computer analysis without access to the original image.

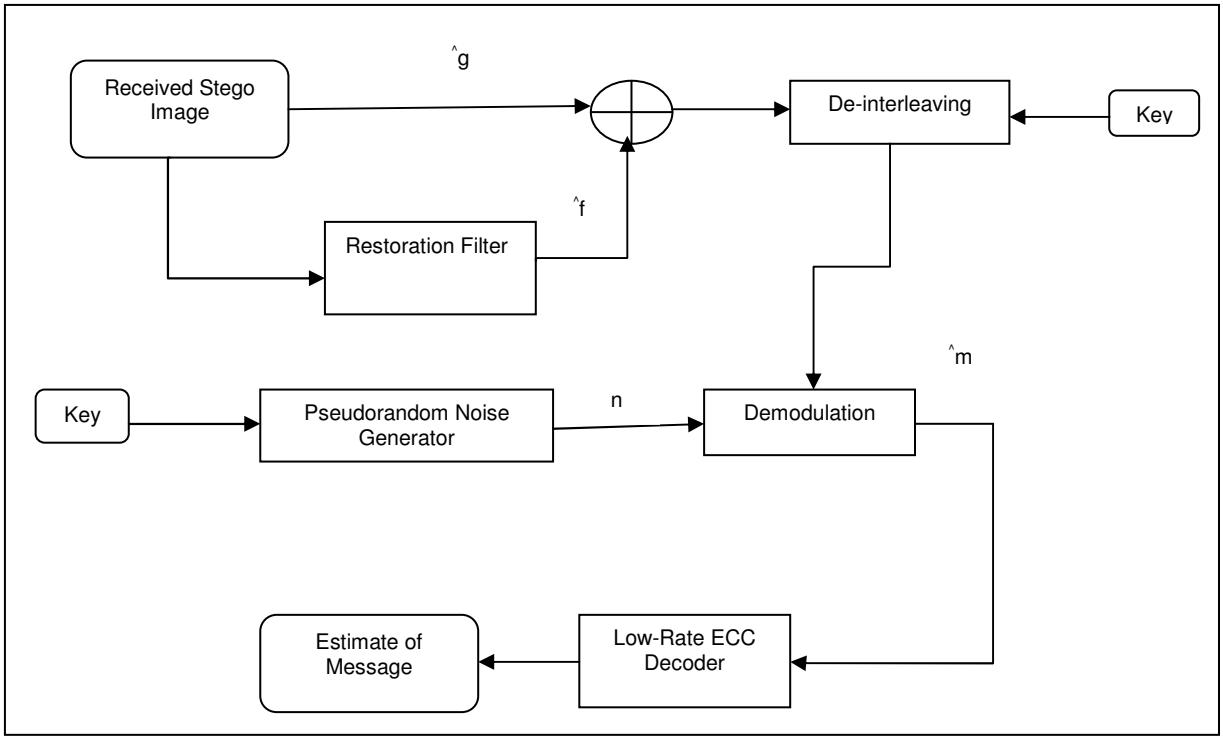


Figure 2.4: Spread Spectrum Decoder

2.2.2.3.3.3 Transform Based Steganography

Another widely used method of steganography is hiding data in the Fourier domain of an image using the Discrete Cosine Transform (DCT), which allows the data to be more resistant to becoming corrupted [17]. The DCT method is widely used over simple steganography methods because hiding data in the Least Significant Bit (LSB) is more easily detectable and data can be lost due to encoding schemes like JPEG. With the DCT method, data can be hidden in a robust format so that the data can be recovered even if some of the image data is lost or modified [22]. Steganography using the DCT is usually accomplished by modulating the size of two or more DCT coefficients within a block of part of an image [19]. The DFT method is similar to the DCT by taking frequencies in the mid-band region, and modifying their coefficients [20]. Images are padded with zeros before taking Discrete Fourier Transform (DFT) to avoid aliasing [21]. Following sections shall give the detailed description for wavelet and contourlet transforms:-

2.2.2.3.3.3.1 Introduction to Wavelet Transform

The history of wavelets is not very old, at most 10 to 15 years, although some of the mathematical background dates back to the theory of Fourier in the nineteen century. The field of wavelets experienced

a fast and impressive start, characterized by a close-knit international community of researchers who freely circulated scientific information and were driven by the researchers' youth enthusiasm.

Fourier set the basis of the frequency which for a long time was the best and the only approach existent in signal analysis. The research gradually moved from frequency based analysis to scale based analysis when the researchers realized that an approach measuring average fluctuations at different scales might prove less sensitive to noise. And so, the wavelet transform was born. The first recorded mention of what we call now a "wavelet" dates back to 1909 in Alferd Haar's thesis [35]. The concept of wavelets in its present theoretical form was proposed later by Morlet. The main algorithm using filters was provided by Stephane Mallat [36] in 1989. Since then, the research has become international. The Mallat algorithm is in fact a classical scheme known in the signal processing community as a two channel subband coder [36].

2.2.2.3.3.3.1.1 Fourier Analysis

Fourier analysis is the most well known tool for signal analysis which breaks down a signal into constituent sinusoids of different frequencies. It transforms our view of the signal from time domain to frequency domain.

Fourier analysis is very useful since in most of the cases the frequency content of a signal is very important. This analysis has a serious drawback that while transforming to frequency domain, time information is lost. Fourier transform cannot tell when a particular event took place. This drawback is not very significant for non-stationery processes i.e., real processes.

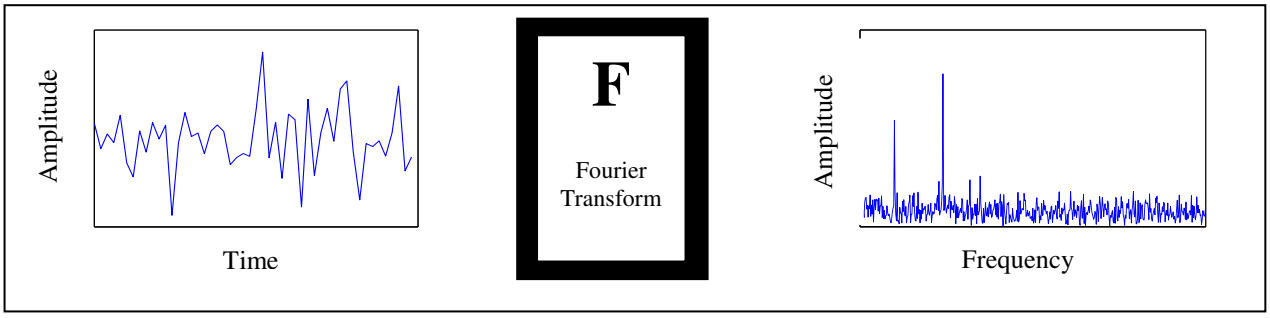


Figure 2.5: Fourier Transform of a Signal

2.2.2.3.3.3.1.2 Short-Time Fourier Transform

To correct this drawback, Dennis Gabor (1946) adapted the Fourier Transform to analyze only a small section of the signal at a time, called windowing the signal. Gabors' adaption, called the Short-Time Fourier Transform (STFT), maps a signal into a two dimensional function of time and frequency.

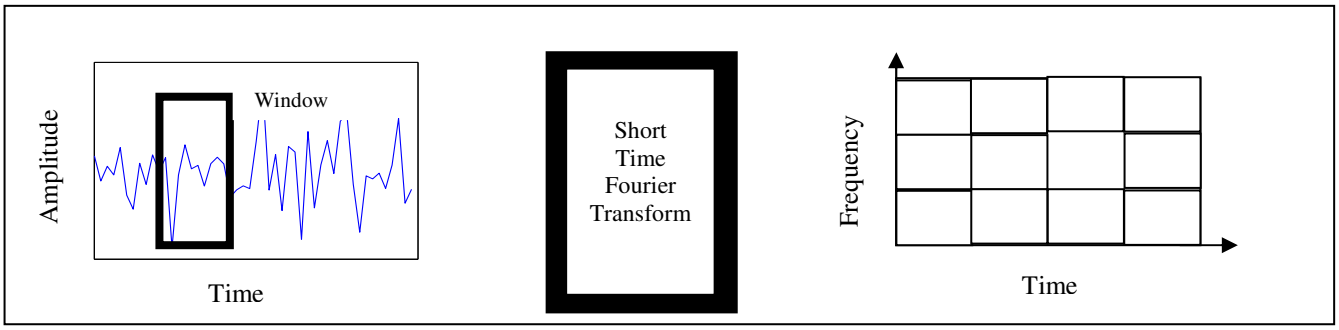


Figure 2.6: Short-Time Fourier Transform of a Signal

The STFT is a compromise between the time domain and the frequency domain views of a signal. It provides some information about when and at what frequencies a signal event occurs with some accuracy depending upon the size of window.

This compromise is very useful but it still has a drawback, once the window size is chosen, it cannot be changed. Many signals require an even more flexible approach, where the window size can be varied in order to determine more accurately either time or frequency.

2.2.2.3.3.3.1.3 Wavelet Analysis

The requirement of varying window size was fulfilled by wavelet analysis which offers a windowing technique with variable sized window. The wavelet transform allows the use of long time intervals where we want more precise low frequency and short intervals where we want high frequency information.

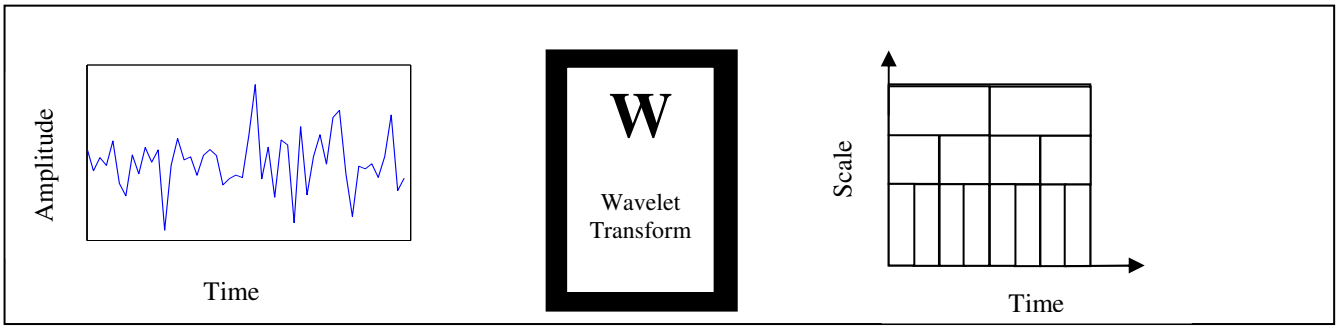


Figure 2.7: Wavelet Analysis of a Signal

One major advantage of wavelet transform is the ability to analyze a localized area of a large signal. In this way, wavelet analysis is capable to reveal aspects of data that other signal analysis technique miss, for example breakdown points trends, discontinuities in higher derivative and self similarity.

2.2.2.3.3.3.1.4 Wavelet Transform

Unlike FFT and DCT, Discrete Wavelet Transform (DWT) is a hierarchical transform which offers the possibility of analyzing a signal at λ different resolutions or levels (with λ integer). The multi resolution nature of the wavelet decomposition compacts the energy in the signal into a small number of wavelet coefficients. For natural images, much of the image energy is concentrated in the LL band that corresponds to the coarsest scale. The LL is not only a coarse approximation of the image but also contains most of the images' energy. It is also statistically observed that the energy in the finer subbands is also concentrated into a relatively a small number of wavelet coefficients. The significant coefficients in the finer subbands do not occur at random, but rather tend to occur in clusters in the same relative spatial location in each of the higher subbands. This self similar hierarchical nature of the wavelet transform can be used to make interband predictions; the location of the significant coefficients in the coarser bands is used to predict the location and magnitude of significant coefficients in finer subbands.

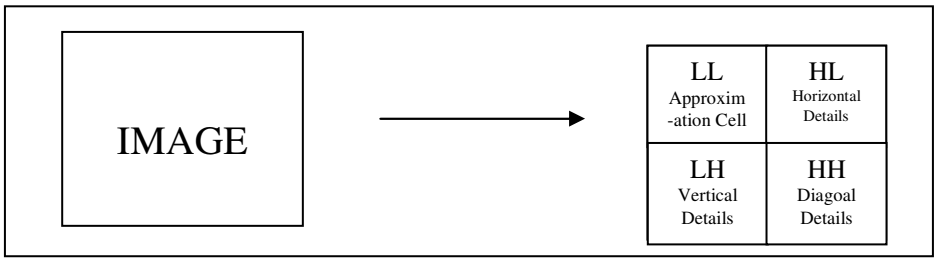


Figure 2.8 2D Wavelet Transform of an Image

2.2.2.3.3.3.1.5 Wavelet Decomposition

Calculating wavelet coefficients at every possible scale is a fair amount of work and it generates a lot of data. It turns out, rather remarkably, that if we choose scales and powers based on powers of two, then our analysis will be much more efficient and just as accurate. An efficient way to implement this scheme using filters was developed by Mallat in 1989 [36]. The Mallat algorithm is in fact a classical scheme known in signal processing community as a "two channel subband coder". This very practical filtering algorithm yields a fast wavelet transform. The filtering process as its most basic level is shown in Figure 2.9.

The original signal I pass through two filtering banks A and D, resulting in approximations and details. The approximations are the high scale, low frequency components of the signal. The details are the low scale, high frequency components. If we actually perform this operation on a real digital signal, we wind up with twice as much data as we started with. Outputs of filters are down sampled to get the required approximation coefficients (cA) and detail coefficients (cD). The decomposition process is shown in Figure 2.10.

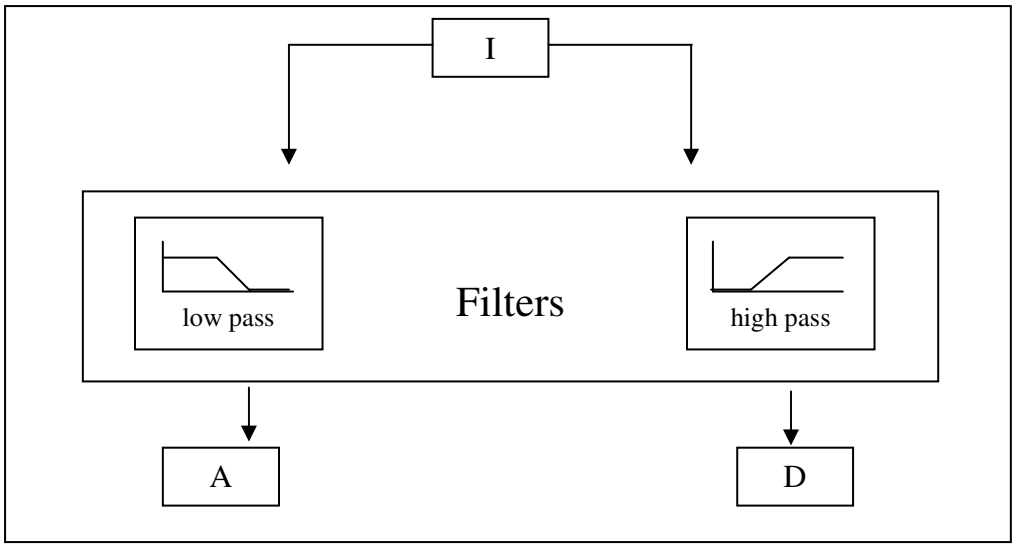


Figure 2.9: Wavelet Decomposition

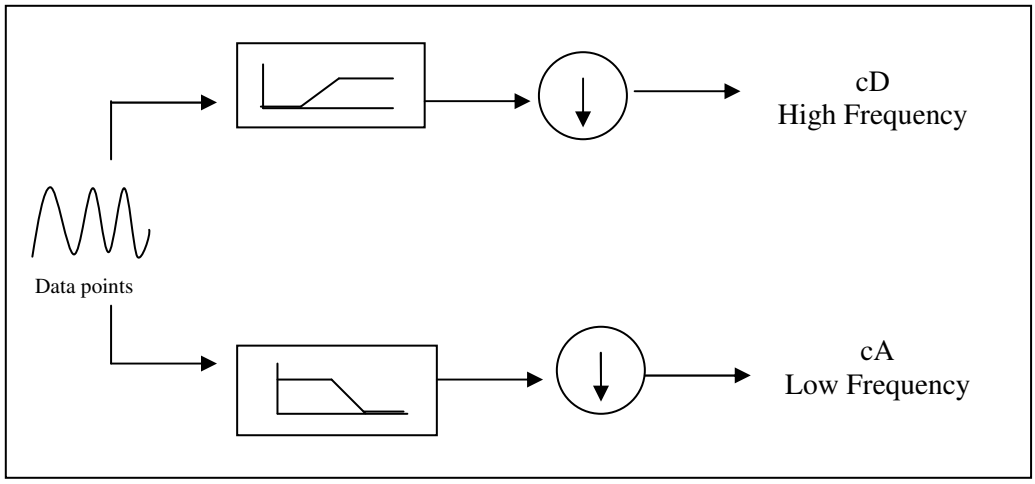


Figure 2.10: Level 1 Wavelet Decomposition

2.2.2.3.3.3.1.6 Wavelet Reconstruction

Down sampling of a signal components performed during the decomposition phase introduces a distortion called aliasing. It turns out that by carefully choosing filters for the decomposition and reconstruction phases that are closely related but still not identical, we can cancel out the effects of aliasing. The low and high pass decomposition filters (L and H) together with their associated reconstruction filters (L' and H'), form a system of what is called quadrature mirror filters.

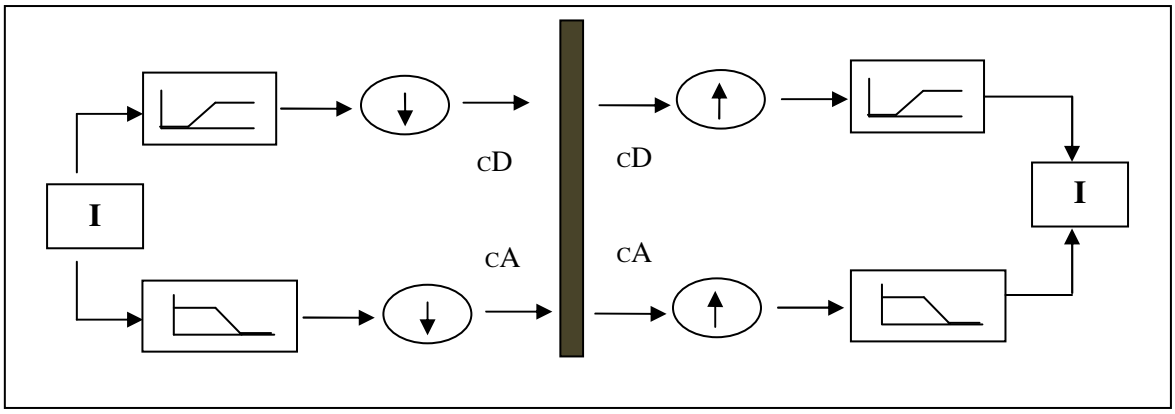


Figure 2.11: Level 1 Wavelet Reconstruction

2.2.2.3.3.1.7 Main Applications of Wavelet Transform

Several thousand papers have been written within last 15 years about the wavelets and their applications. This proves once more the success of wavelets. Steganography tends to become another successful application area. Probably one of the most popular applicants of wavelet is the FBI fingerprints and JPEG 2000 format.

Medicine is very prolific application field of wavelets; especially in heart diagnosis EKG/ECG – (Electrocardiography), EEG (Electroencephalography), mammography and MRS (Magnetic Resonance Spectra).

Wavelets are well-suited for approximating data with sharp discontinuities. The wavelet analysis procedure is to adopt a wavelet prototype function, called an analyzing wavelet or mother wavelet. Temporal analysis is performed with a dilated, low frequency version of the same wavelet. Because the original signal or function can be represented in terms of a wavelet expansion (using coefficients in a linear combination of the wavelet functions), data operations can be performed using just the corresponding wavelet coefficients. And if you further choose the best wavelets adapted to your data, or truncate the coefficients below a threshold, your data is sparsely represented.

Recently many researchers perform quality of work to improve the image quality and capacity in this domain. Focus was made on Discrete Cosine Transform (DCT) domain which also proved remarkable than spatial domain techniques.

Embedding in DCT domain is simply done by altering the DCT coefficients, for example by changing the least significant bit of each coefficient. One of the constraints of embedding in DCT domain is that many

of the 64 coefficients are equal to zero, and changing zeros to non-zeros values will have an effect on the compression rate. That is why the number of bit one could embed in DCT domain, is less that the number of bits one could embed by the LSB method. Also the embedding capacity becomes dependent on the image type used in the case of DCT embedding, since depending on the texture of image the number of non-zero DCT coefficients will vary. Although changing the DCT coefficients will cause unnoticeable visual artifacts, they do cause detectable statistical changes.

K B Shiva Kumar , K B Raja, R K Chhotaray, Sabyasachi Pattanaik in [58] proposed a Bit Length Replacement Steganography Based on DCT Coefficients in which a cover image is segmented in 8x8 blocks and then Discrete Cosine Transform is executed on each segmented block. The Most Significant Bits of message to embed are used with DCT coefficients of image. It resulted in better PSNR and capacity.

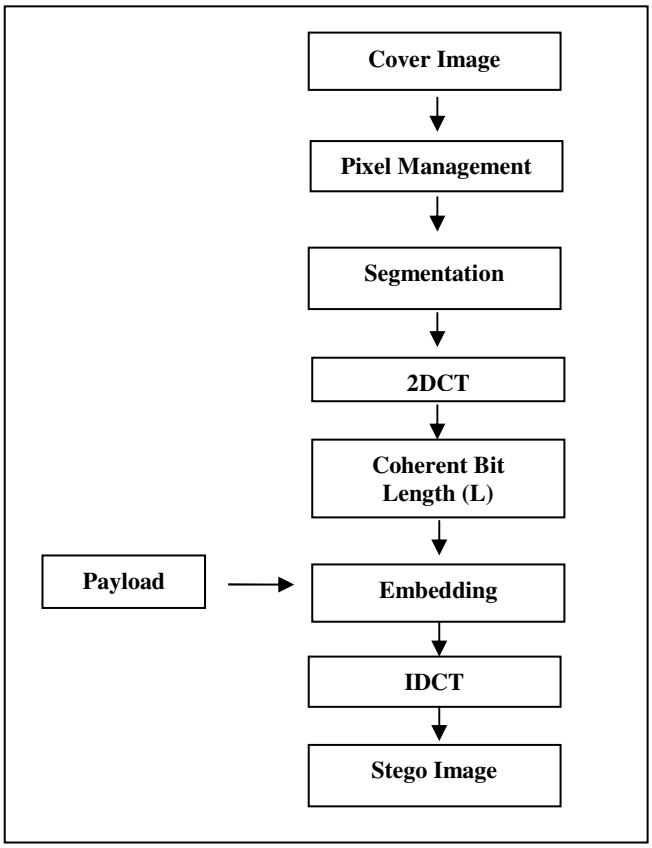


Figure 2.12 Block diagram of [58]

Amitava Nag, Sushanta Biswas, Debasree Sarkar & Partha Pratim Sarkar in [59] presented a novel technique using DWT transform for the cover image transformation and then Huffman is used on secret message before embedding. It uses only high frequency coefficients for embedding message bits and neglected low ones to get better image quality. Figure 2.13 shows the embedding scheme in more detail.

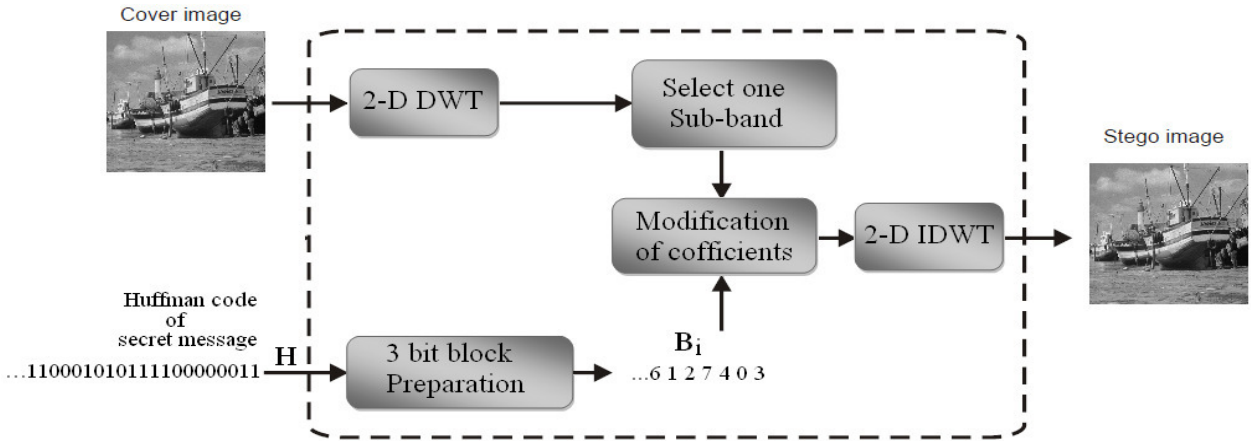


Figure 2.13 Block diagram of [59]

M. F. Tolba, M. A. Ghonemy, I. A. Taha, and A. S. Khalifa in 2004 [57] utilizes wavelet transforms that map integers to integers and proposed an algorithm that embeds the message bitstream into the LSB's of the integer wavelet coefficients of a true-color image. Experimental results showed the high invisibility of the proposed model even with large message size. The embedding algorithm is shown in Figure 2.14

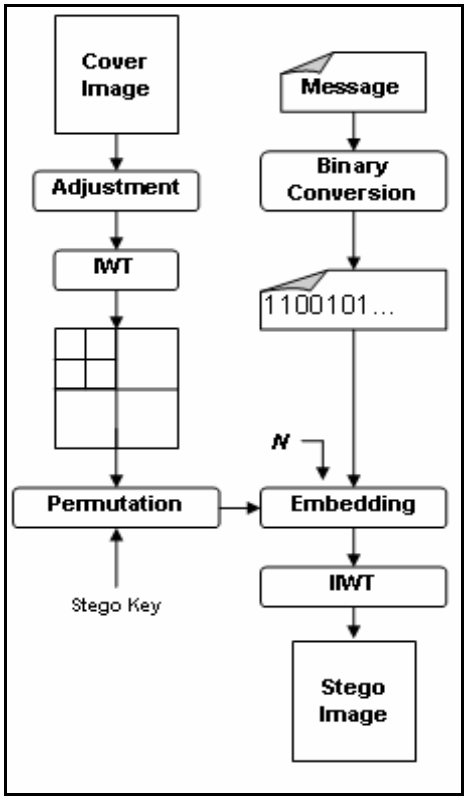


Figure 2.14 Block diagram of [57]

Bo Yang and Beixing Deng in 2006 [60] presented an image steganographic scheme for gray scale images. This scheme hides a small gray size image in a large one. Input image is first transformed using

Arnold transformation to preserve large area information in cover image. Arnold is a tool converting one matrix into another. Suppose A be MxM matrix, a point (i0,j0) can be shifter to (i,j) by using Equation 2.1.

$$\begin{bmatrix} i0 \\ j0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} \ (Mod\ M) \qquad \text{Equation 2.1}$$

DWT is then performed on cover image in which small image is encoded and also on Arnold transformed image. Next step performs the bit streaming of DWT coefficients to embed the approximate component of secret image into the approximate component of cover image using LSBs. This work also claimed that original image is not required in extraction process.

2.2.2.3.3.3.2 Introduction to Contourlet Transform

The contourlet transform was recently proposed to overcome the limited ability of wavelet to represent image edges and discontinuities. Besides retaining the desirable characteristics of wavelet transform, such as multi-resolution and localization, it has two additional important features: directionality and anisotropy.

2.2.2.3.3.3.2.1 Contourlets

In particular, Contourlets have elongated supports at various scales, directions, and aspect ratios. This allows Contourlets to efficiently approximate a smooth contour at multiple resolutions.

2.2.2.3.3.3.2.2 Contourlet Transform

Recent researches' have identified that the main cause of inefficiency is the lack of directionality and anisotropy and also described bi-dimensional smooth contours, and how new transforms, instead, aspire to these features. In the last years a lot of different transforms have been proposed (contourlets [37], directionlets [38], curvelets [39], bandelets [40], etc.), that overcome wavelet limits in representing images contours. The results are mostly theoretical, but they are quite promising, and stimulate the quest for actual coding algorithms based on these new tools.

We chose the contourlet transform [37] for several reasons: it has a number of desirable characteristics (directionality, anisotropy, etc.) and an almost optimal non linear approximation behavior for simple classes of images; in addition, unlike other transforms, it is easily implemented by a filter bank. Its main

drawback is a slight redundancy which, however, is not really a problem in the context of low bit-rate coding [41].

Contourlet transform was introduced by Do and Vetterli in 2005 [37], it comprises two blocks, a Laplacian pyramid and a directional filter bank (DFB). The Laplacian pyramid (LP) was proposed by Burt and Adelson [42] in 1983 as a multi-resolution image representation. In the first stage of the decomposition, the original image is transformed into a coarse signal by mean of a low pass filtering and a down sampling. This coarse version is then up-sampled and filtered to predict the original image (Figure 2.15 [43]). This procedure can be repeated iteratively in order to obtain a multi-resolution decomposition.

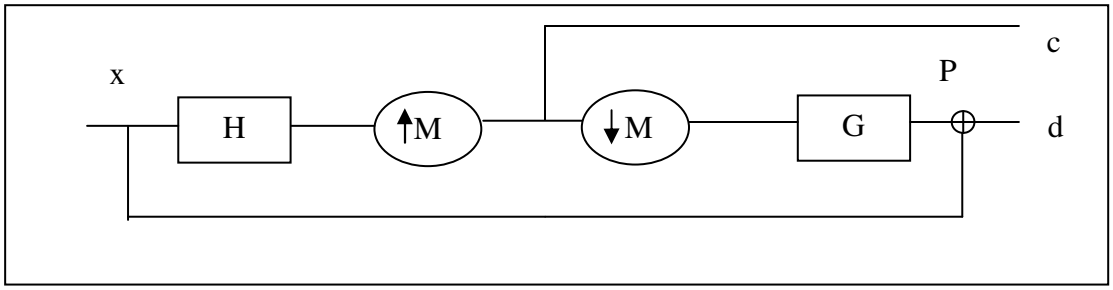


Figure 2.15: Laplacian Pyramid (LP)

2.2.2.3.3.3.2.3 Why Contourlet Transform?

LP decomposition is a redundant representation so it is natural to wonder why in contourlet it has been preferred to critically sampled filter banks as discrete wavelet transforms. The motivation must be detected in the successive use of Laplacian pyramid; in fact, in contourlet decomposition, a directional filtering is performed on the bandpass versions of input signal. So it needs a decomposition that permits further subband decomposition of its bandpass images. To this target the LP has two advantages over the critically sampled wavelet scheme: first, it generates only one bandpass version, second, it does not suffer from the frequencies "scrambling". This problem arises in the critical sampling filter banks when, because of down sampling, the high pass channel is folded back into the low frequency band and its spectrum is reflected [44]. This problem is overcome in LP by downsampling only the low pass channel. A peculiarity of LP used in contourlet transform is the reconstruction structure.

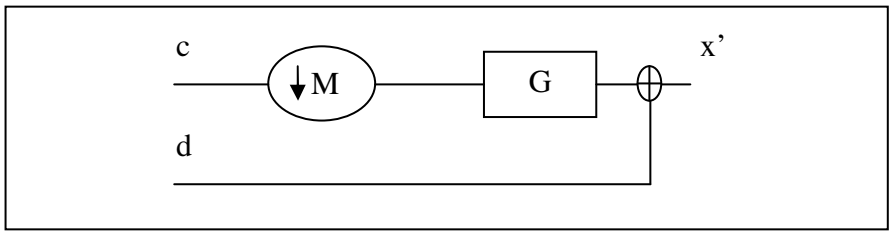


Figure 2.16: Simple Reconstruction Scheme for Laplacian Pyramid (LP)

In fact, most applications employ the simple synthesis operator shown in Figure 2.16 to reconstruct the image from the LP, but this simple synthesis operator is not optimal in terms of minimizing the distortion propagation from the subbands of the LP to the reconstructed image.

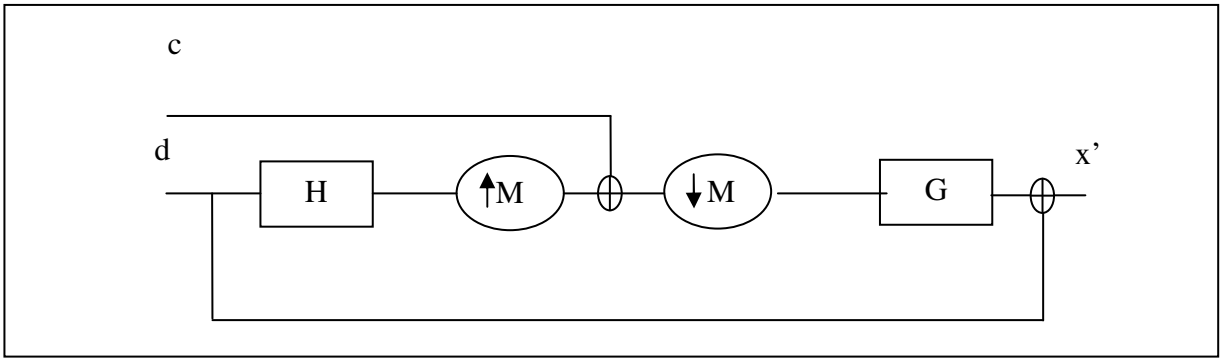


Figure 2.17: Do Vetterli Reconstruction Scheme for Laplacian Pyramid (LP)

In contourlet scheme, Do and Vetterli use, instead, a structure which implements the dual frame reconstruction (Figure 2.17) (LP multiresolution pyramid is in practice a frame expansion), because this is the optimal choice in presence of noise [43]. The second block of contourlet decomposition is a directional filter bank that singles out directional components, with a number of directions that can vary as a power of two.

Bamberger and Smith ([45]) introduced a perfect reconstruction directional filter banks (DFB), which can be maximally decimated, implemented via an l-level tree-structured decomposition that leads to 2^l subbands with wedge-shaped frequency partition. Figure 2.14 shows an example of DFB frequency partitioning with l = 3, the subbands 0-3 correspond to the mostly horizontal directions, while subbands 4-7 correspond to the mostly vertical directions.

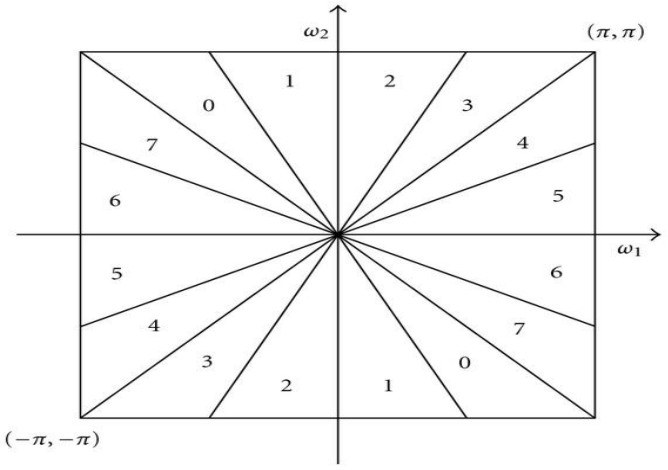


Figure 2.18: Directional Filter Bank Frequency Partitioning with l=3

In conclusion the entire scheme of contourlet transform is shown in Figure. 2.19. Theoretically, the number of directions in which one can divide the bandpass subbands at each level of decomposition is a free parameter, but, to make contourlet basis to be anisotropic, as well as directional, a condition must be imposed.

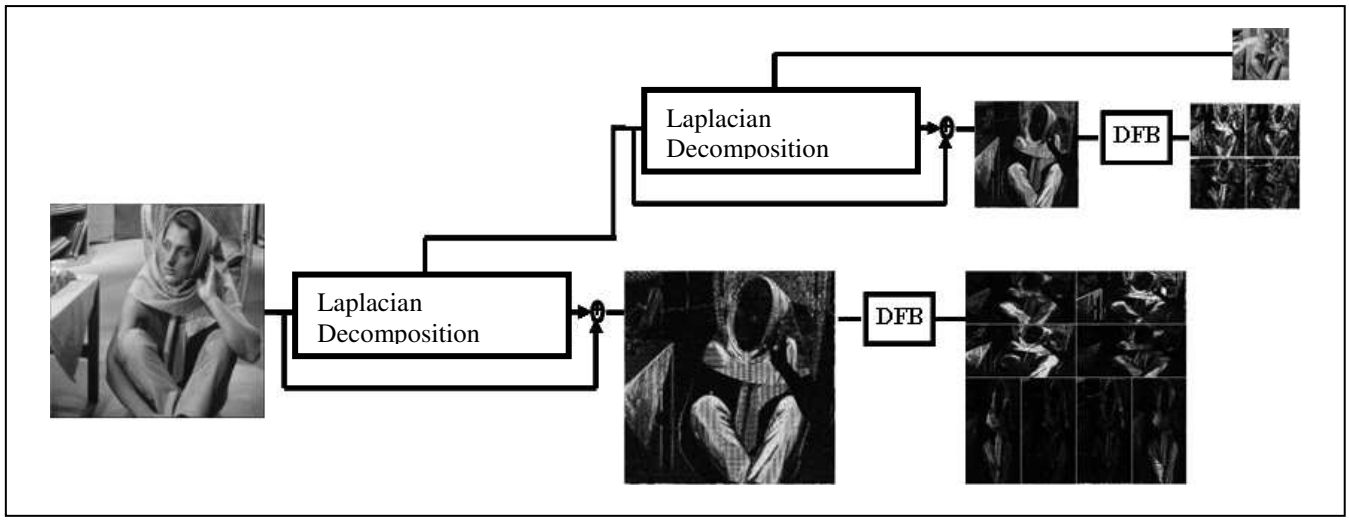


Figure 2.19: Contourlet filter bank [45]

Hedieh Sajedi, Mansour Jamzad in 2009 [56] proposed a method called ContSteg to embed the information in 4x4 blocks of contourlet subbands by exchanging the value of two coefficients in a block of contourlet coefficients. The embedding algorithm of ContSteg is shown in Figure 2.20

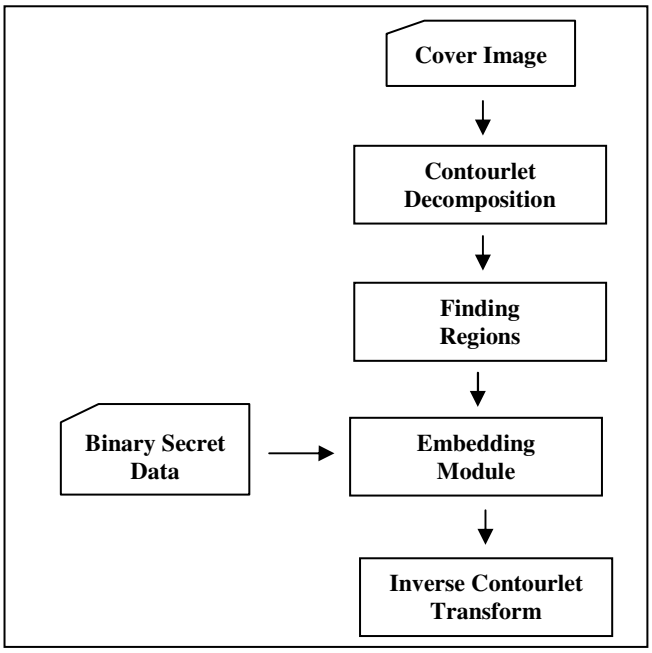


Figure 2.20 Block diagram of [56]

Hedieh Sajedi, Mansour Jamzad in 2010 [61] presented a contourlet based image steganographic method to hide secret information in a cover image. Presented work decomposed cover image using contourlet transform and then embed information by increasing and decreasing the value of a coefficient in a subband obtained by contourlet decomposition. This paper presented the comparative study of effect of cover image on embedding and suggested that choosing proper cover image would decrease the chances of detectability.

2.2.2.4 Gray Scale Image Steganography

2.2.2.4.1 Labeling Method

Motameni, Norouzi, Jahandar, and A. Hatami in their paper "Labeling Method in Steganography" [23] propose a method of hiding information in gray scale images using labeling. The method first finds the binary value for each character in text and then finds dark places in the gray image by converting input image to binary image on 8-connectivity basis. Then the image is converted to RGB image in order to find the dark places. In this way, each sequence of gray color turns into RGB color. If the Gary image is very light, the histogram must be changed manually to find just dark places. In the final stage each 8 pixels of dark places are considered as a byte and binary value of each character is put in low bit of each byte that was created manually by dark places pixels for increasing security of the main way of steganography.

2.2.2.4.2 Hiding Color image in a Gray Scale Image

Chaumont and W. Puech propose a method to embed the color information of an image in a corresponding grey-level image [24]. The objective of this work is to allow free access to the grey-level image and give color image access only if you own a secret key. This method is made of three major steps which are the color quantization, the ordering and the data hiding.

2.2.2.5 Color Image Steganography

2.2.2.5.1 Using Edge Detection

In this method, edge pixels are selected in order to hide the data. One common way to hide the data is "Least Significant Bit Insertion". This method modifies the low order bit of each pixel to match the message to hide. The selection of pixels in which the message will be embedded is very important because modified pixels in areas of the image where there are pixels that are most like their neighbors are much more noticeable to the naked eye. A single modified pixel stands out among its uniform neighbor pixels thus making the image suspicious. One possible solution for this problem is to select the edge-

pixels of the image to hide the message [25]. It is not noticeable when a single pixel is modified when its surrounding pixels are least like it.

2.2.2.5.2 Using Variable-Bits

This paper presents an algorithm for storing variable number of bits in each channel (R, G or B) of pixel based on the actual color values of that pixel: lower color component stores higher number of bits [26]. The procedure followed is as follows:-

1. One of the Red, Green and Blue channels is used as an indicator channel. This has to be agreed upon by the sender and receiver prior to communication.
2. Data is stored in either of the other two channels depending on their intensity values. The channel that has the lowest color value will embed data in LSBs.
3. To retrieve data, the receiver analyzes the least significant bits of the other two channels. If the last two bits are same then the channel following the indicator channel stores the data. Otherwise the channel preceding the indicator channel contains the embedded data.

2.2.2.5.3 Using differential phase-shift keying techniques

Here following steps are used to achieve steganography:-

a. Compress the secret image to reduce the number of secret bits. The set partitioning in hierarchical trees (SPIHT) codec were used to obtain a high reconstructed image quality and low bit rate image compression.
b. A neighbor block signal phase comparison (NBSPC) mechanism is used to offer the location for secret data embedding.
c. A fold phase distribution differential phase-shift keying FPDPSK mechanism is used to improve the quality of the cover image [27]

2.2.2.5.4 Using best-block matching and k-means clustering

An image-hiding technique using block-matching procedure is proposed in [28]. The method suggests to:-

a. Split the entire image into multiple non-overlapping blocks.
b. For each block, best matching block is searched from a series of numbered candidate blocks [generated from the cover image.]
c. The blocks that are not well matched, a *k*-means clustering method is applied to determine some representative blocks.
d. These representative blocks are used to replace some of the candidate blocks that were not referenced in the block matching step.

Finally, the obtained indices of blocks are encoded using Huffman coding scheme, and then recorded in the LSB Channels of the cover image.

2.3 Detailed Look At Encryption Algorithms

Data encryption algorithms can be classified in several ways. They can be categorized based on number of keys employed for encryption and decryption. There are three types of encryption algorithms:-

a. **Secret Key Algorithms:** Use a single/same key for encryption and decryption.
b. **Public Key Algorithms:** Use different keys for encyption and decryption.
c. **Hash Functions:** Use a mathematical transformation to irreversibly encrypt data.

Figure 2.21 shows the above in more detail.

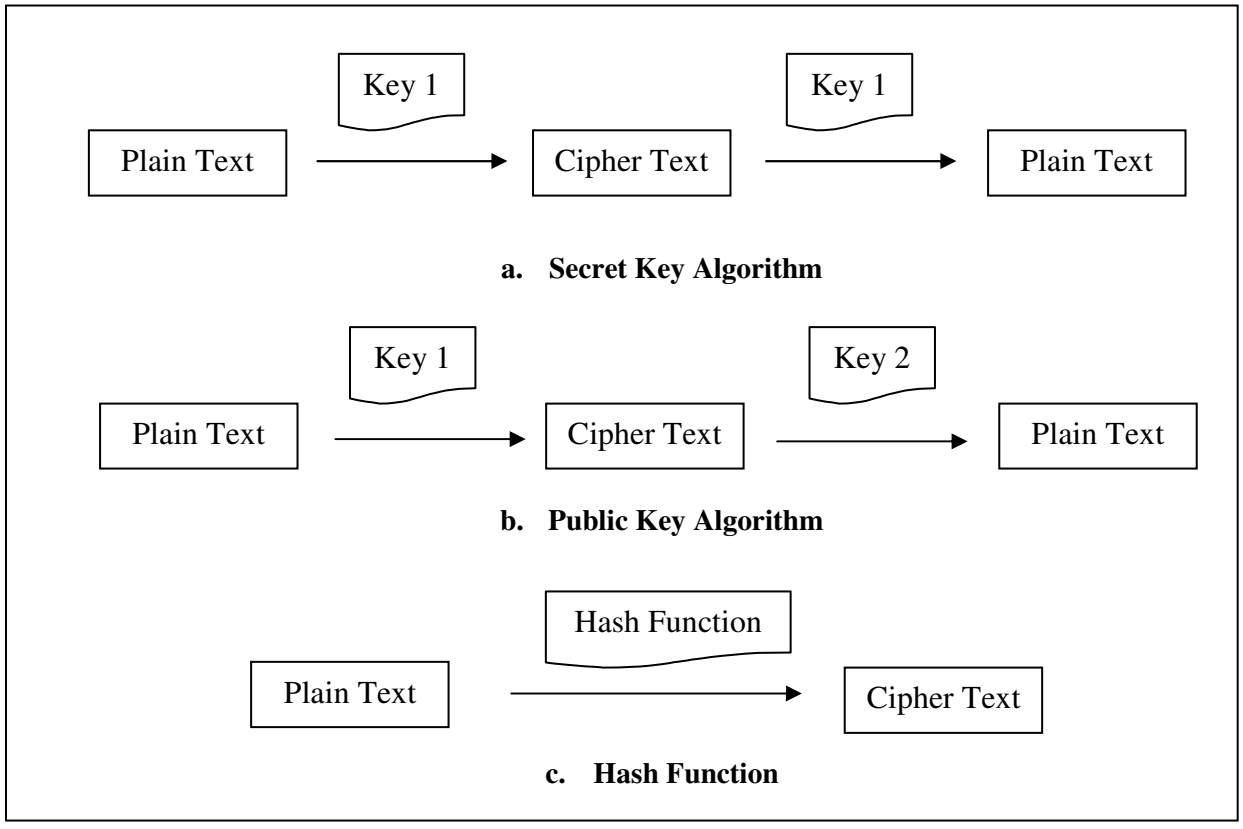


Figure 2.21: Encryption Algorithms

Further is the description of different encryption algorithms are covered along with there performance evaluation.

2.3.1 Secret Key Encryption Algorithms

The well-known secret key encryption algorithms are:-

a. DES
b. Triple DES
c. AES (Rijndael)
d. Blowfish

2.3.1.1Data Encryption Standard (DES)

DES (Data Encryption Standard) is currently the most widely used block cipher in the world. In May 1973, NIST (then NBS) called for possible encryption algorithms for use in unclassified government applications. As a response to this, IBM submitted Lucifer algorithm, which was later redesigned to become the Data Encryption Standard (DES). In November 1976, DES was adopted as a US federal standard. DES was published by NBS as a hardware only scheme in January 1977, and by ANSI for both hardware and software in ANSI X3.92-1981 and X3.106-1983 [29].

Subsequently, DES became the widely adopted encryption algorithm and is in many standards around the world (e.g. Australian Standard AS2805.5-1985). One of the largest users of the DES is the banking industry. It is for this use that the DES was primarily standardized, with ANSI reconfirming its use for 5 year periods – in future it will be replaced with AES.

Although the DES standard is public, the design criteria used are classified. There has been considerable controversy over the design, particularly in the choice of a 56-bit key [29].

2.3.1.1.1DES Security

Recent analysis has shown that the choice of 56-bit key for DES was appropriate, and that DES is well designed. However, rapid advances in computing speed though have rendered the 56-bit key susceptible to exhaustive key search, as predicted by Diffie & Hellman [30]; DES has now been demonstrated to be breakable:-

a. In 1997, DES was broken by using a large network of computers in a period of few months
b. In 1998, EFF broke DES using dedicated hardware in just a few days [33]
c. In 1999, DES was broken in just 22 hours

DES has also been theoretically broken using Differential or Linear Cryptanalysis [31]. However, this is not yet a practical problem unless the information needs to be secured for long periods of time. The key

issue here is how much is the worth of security. If it were required only to secure a transfer of funds for a few seconds, it would hardly matter if the message were broken after the transaction has completed. However, a highly confidential spy report is to be encrypted, it would not be desirable for it to be broken even 50 years later – and thus DES would not be used for this type of application.

2.3.1.1.2Overview of the DES Encryption Algorithm

The basic process in enciphering a 64-bit data block using the DES (Figure 2.22) consists of:-

a. an initial permutation (IP)
b. 16 rounds of a complex key dependent calculation f
c. a final permutation, being the inverse of IP

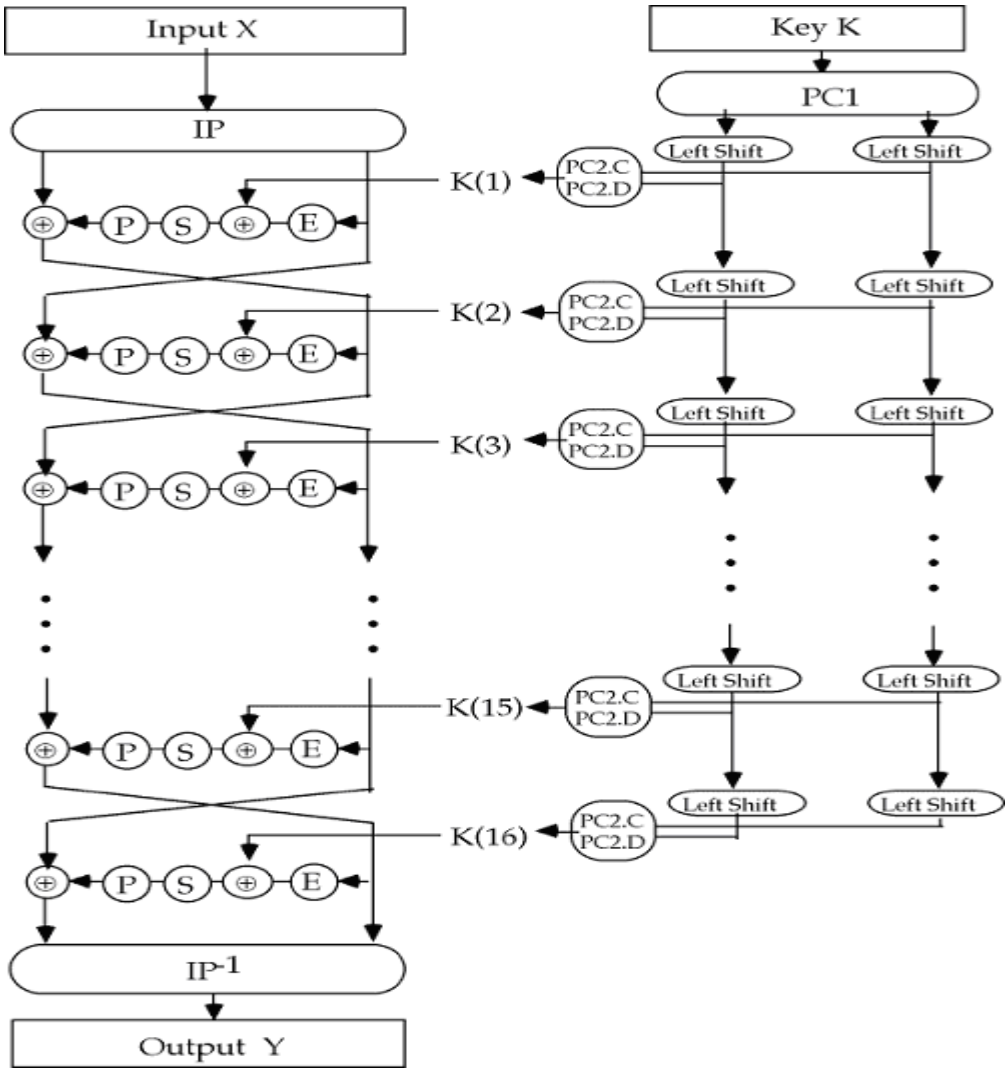


Figure 2.22: DES Encryption

DES is also a classic Feistel cipher, with 16 rounds, working on 64-bit blocks of data. Its structure and internal workings will be examined in detail, because it is both a widely used and known cipher, and also because it embodies many of the concepts of modern block ciphers.

DES Key Schedule

The subkeys used in each round are formed by the key schedule which consists of an initial permutation of the key (PC1) which selects 56-bits of key from 64 bits, and divides them into two 28-bit halves. Each of the sixteen rounds consists of:-

a. selecting 24-bits from each half
b. permuting them by PC2 for use in function f,
c. rotating each half separately either one or two places depending on the key rotation schedule KS

This can be described functionally as:-

$SK_i = PC2(KS(PC1(Key),i))$

where i is round number and KS is key rotation schedule

It should be noted that following PC1, the key schedule is in two independent halves, the rotations happen separately on each half, and PC2 is effectively two smaller permutations taking bits strictly from C to S-boxes 1-4, and from D to S-boxes 5-8 (Figure 2.17).

PC1

PC1 is used to select 56 of 64 bits supplied as the key. Every 8th bit is discarded (assumed to be parity). PC1 also splits the key bits into 2 halves (C and D). In DES, bits are numbered from left, i.e. most significant bit is numbered as 1. The C and D halves consist of following bits each:-

57, 49, 41, 33, 25, 17, 9, C Half
1, 58, 50, 42, 34, 26, 18,
10, 2, 59, 51, 43, 35, 27,
19, 11, 3, 60, 52, 44, 36,
63, 55, 47, 39, 31, 23, 15, D Half
7, 62, 54, 46, 38, 30, 22,
14, 6, 61, 53, 45, 37, 29,
21, 13, 5, 28, 20, 12, 4

The 56 bit size comes from security considerations as it is now known. It was considered big enough so that an exhaustive key search was about as hard as the best direct attack (a form of differential cryptanalysis called a T-attack, known by the IBM & NSA researchers), but no bigger. The extra 8 bits were then used as parity (error detecting) bits, which makes sense given the original design use for

hardware communication links. Note that the bit numbering for DES reflects IBM mainframe practice, and is the opposite of what is now mostly used. The key schedule for PC1 should be read as follows: Bit 1 (leftmost) of PC1 comes from bit 57 of the input; bit 2 from bit 49; bit 3 from bit 41 etc. to bit 56 (rightmost) from bit 4.

PC2

PC2 is used to select two lots of 24 bits each from the 56 bits selected in PC1. This schedule is used in each round of the DES data computation. The C half provides bits to S1-S4, while the D half provides to S5-S8. The PC2 key schedule is as below:-

14, 17, 11, 24, 1, 5,	C half
3, 28, 15, 6, 21, 10,	(bits 1-28)
23, 19, 12, 4, 26, 8,	
16, 7, 27, 20, 13, 2,	
41, 52, 31, 37, 47, 55,	D half
30, 40, 51, 45, 33, 48,	(bits 29-56)
44, 49, 39, 56, 34, 53,	
46, 42, 50, 36, 29, 32	

PC2 collects groups of bits from across each half to feed into each S-box, thus making sure the output depends in a complex way on the key. The split into 2 halves again reflects the limitations of h/w at the time. Note that 4 bits aren't used each time.

Key Rotation Schedule

The key rotation schedule KS is specified as below: -

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
KS	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1
Total	1	2	4	6	8	10	12	14	15	17	9	21	23	25	27	28

The key rotation schedule along with PC2 ensures that on successive rounds any given key bit affects a different S-box, and if missed for one round, it will be used in the next. Also, after sixteen rounds each key half register is back to its starting value, which makes computing keys for decryption easier, just rotate the other way, and use the rotation amounts in the reverse order.

DES Data Computation

In this section, the working of DES computation rounds is explained, which provide the actual scrambling (encryption) of the data. First of all, the input data undergoes an initial permutation, which rearranges the input, then 16 rounds of the complex non-linear round function f, which performs expansion E, key addition, substitution S, and a permutation P. The result of round function is XOR'ed with left half of data (LH), and the halves are swapped. This can be described functionally as below: -

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \text{ XOR } P(S(E(R_{i-1}) \text{ XOR } SK_i))$$

Figure 2.23 shows a single round of DES computation. At the end of sixteen rounds, a final permutation (inverse of initial permutation) is performed.

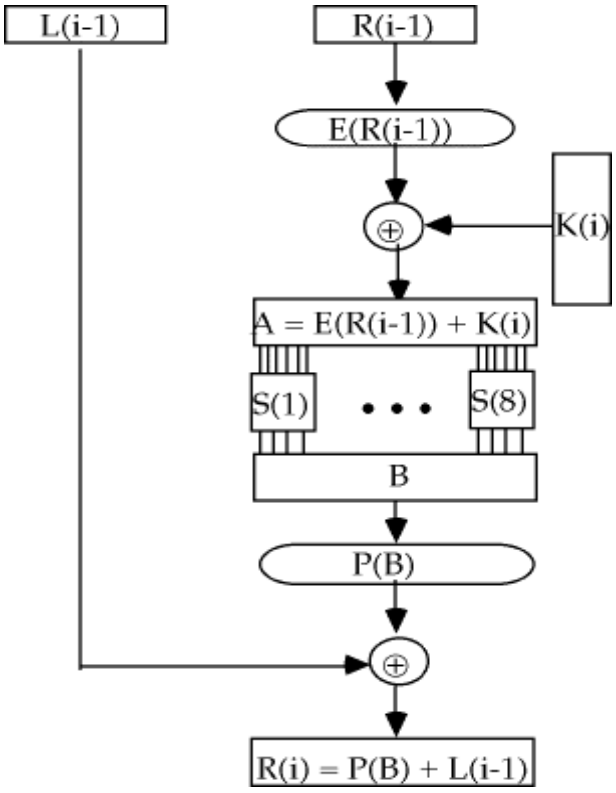


Figure 2.23: A single round of DES encryption

Initial Permutation (IP)

It is the first step of the DES data computation. It reorders the input data bits in such a way that all even bits are in left half (LH), and all odd bits are in right half (RH). This does not, generally, improve actual security but it makes the cipher more complex: The IP is described below:-

58, 50, 42, 34, 26, 18, 10, 2,
60, 52, 44, 36, 28, 20, 12, 4,
62, 54, 46, 38, 30, 22, 14, 6,
64, 56, 48, 40, 32, 24, 16, 8,
57, 49, 41, 33, 25, 17, 9, 1,
59, 51, 43, 35, 27, 19, 11, 3,
61, 53, 45, 37, 29, 21, 13, 5,
63, 55, 47, 39, 31, 23, 15, 7

For example, IP (675a6967 5e5a6b5a) = (ffb2194d 004df6fb)

Expansion Function

The expansion function E expands right half RH of the input data from 32 to 48 bits by duplicating some of the bits. More specifically, the input is split into 8 groups of 4 bits each, and then bits are duplicated from either side to form groups of 6 bits. The expansion function is described by the table below:-

32, 1, 2, 3, 4, 5,
4, 5, 6, 7, 8, 9,
8, 9, 10, 11, 12, 13,
12, 13, 14, 15, 16, 17,
16, 17, 18, 19, 20, 21,
20, 21, 22, 23, 24, 25,
24, 25, 26, 27, 28, 29,
28, 29, 30, 31, 32, 1

For example, E (004df6fb) = 20 00 09 1b 3e 2d 1f 36

The expansion function expands the data input to 48-bits, before it is added to the subkey. The duplicated outside bits in each group of 6 bits act as a key to the following S-box, allowing the data (as well as the key) to influence the S-box operation. This is known as autokeying or autoclave.

Key Addition

The expanded 48 data bits are then XOR'ed (addition modulo 2) to the 48 bit subkey selected by PC2. Thus, the key bits influence the result of the round function.

Substitution Boxes

There are 8 S-boxes, each of which maps 6 bits to 4 bits. Each S-box actually consists of four little 4-bit boxes. To apply an S-box operation on a 6-bit input, outer bits 1 & 6 (row bits) are used to select one of the 4 rows of the S-box, while inner bits 2-5 (column bits) are used to select a column of the S-box. The output of S-box operation is the 4-bit number at row and column thus identified. The S-boxes are as below:-

0 1 2 3 4 5 6 7 8 9 a b c d e f

S[1]

14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7
0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8
4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0
15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13

S[2]

15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10
3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5
0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15
13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9

S[3]

10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8
13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1
13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7
1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12

S[4]

7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15
13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9
10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4
3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14

S[5]

2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9
14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6
4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14
11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3

S[6]

12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11

10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8
9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6
4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13

S[7]

4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1
13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6
1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2
6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12

S[8]

13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7
1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2
7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8
2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11

As an example, S(18 09 12 3d 11 17 38 39) = 5fd25e03

It is to be noticed that the values in the tables above are given in decimal (as is usually shown in the specs). When working through the examples, the decimal values need to be converted to binary or hexadecimal.

Permutation Operation

The permutation operation P diffuses the S-box outputs across different S-box inputs in the next round. As a result, the output appears scrambled, but does have some pattern. The permutation operation ensures each S-box output affects both row and column bits. The permutation operation is defined as below:-

16, 7, 20, 21,
29, 12, 28, 17,
1, 15, 23, 26,
5, 18, 31, 10,
2, 8, 24, 14,
32, 27, 3, 9,
19, 13, 30, 6,
22, 11, 4, 25

For instance, P (5fd25e03) = 746fc91a

Addition to Left Half and Swap

Once the result of the round function f has been computed, it is XOR'ed (added modulo 2) to the left half LH data, and the two halves are exchanged (except in the last round).

Final Permutation

This is the last step in DES encryption, and is performed after 16 rounds. The result of final permutation becomes the output from the DES computation. It is exact inverse of the initial permutation (IP). Hence, when decrypting, the IP operation undoes this operation. The final permutation is defined by the following table:-

40, 8, 48, 16, 56, 24, 64, 32,
39, 7, 47, 15, 55, 23, 63, 31,
38, 6, 46, 14, 54, 22, 62, 30,
37, 5, 45, 13, 53, 21, 61, 29,
36, 4, 44, 12, 52, 20, 60, 28,
35, 3, 43, 11, 51, 19, 59, 27,
34, 2, 42, 10, 50, 18, 58, 26,
33, 1, 41, 9, 49, 17, 57, 25

For example, FP(068dddcd 1d4ccebf) = 974affbf, 86022d1f

2.3.1.2Triple DES (3DES)

The triple DES (3DES) algorithm was needed as a replacement for DES due to advances in key searching. 3DES is a proposal based on the existing DES, and was standardized in ANSI X9.17 & ISO 8732 and in PEM for key management. It was also proposed for general EFT standard by ANSI X9 [29]. It is backwards compatible with existing single DES (when K1=K2=K3). The 3DES algorithm uses either two or three 56-bit keys. Thus the effective key length is up to 168 bits. 3DES is defined by the following function:-

$$C = DES_{K3}\{DES^{-1}_{K2}\{DES_{K1}(P)\}$$

where P = Plaintext
C = Ciphertext
DES_K = DES encryption using key K
DES^{-1}_K = DES decryption using key K

2.3.1.2.1 Security of Triple DES

There are no known practical attacks on 3DES, and a brute force attack is also practically impossible due to the large size of key. Meet-in-the-middle attacks need 2^ (56) plaintext-ciphertext pairs per key. Because of these benefits, 3DES is a popular current alternative to the DES. However, its major disadvantage is its slow execution speed (almost 3 times slower than DES).

2.3.1.2.2 Modes of Use

Block ciphers encrypt a fixed size block of information, for example DES encrypts 64-bit blocks of data, using a 56-bit key. Given that the size of information that needs to be encrypted is arbitrary (which may not be in 64-bit blocks), there is a need for some way of specifying how to use DES in such cases [34].

The way in which a block cipher is used is called its *Mode of Use*. For DES, four modes have been defined in ANSI standard *ANSI X3.106-1983 Modes of Use*, i.e. Electronic Codebook Mode, Cipher Block Chaining Mode, Cipher Feedback Mode, and Output Feedback Mode. The first two modes are called block modes, while the last two are stream modes. Block modes operate on blocks of data (i.e. the input data is divided into blocks and then each block is encrypted). On the other hand, stream modes operate on bits or bytes of a message stream. These modes are described below:-

DES (or any block cipher) forms a basic building block, which encrypts a fixed sized block of data. However to use these in practice, arbitrary amounts of data need to be handled usually, which may be available in advance (in which case a block mode is appropriate), or may only be available a bit/byte at a time (in which case a stream mode is used).

2.3.1.2.2.1 Electronic Codebook (ECB) Mode

In this mode, the message is broken into independent blocks, which are encrypted. Each block is a value which is replaced with its cipher, like a codebook, hence the name. In ECB, each block is independent of all others. Functionally, this can be described as:-

$$C_i = DES_{K1}(P_i)$$

where P_i is the *ith* block of Plaintext

C_i is the *ith* block of Ciphertext

ECB is the simplest of the modes, and is usually used when only a single block of data needs to be sent (e.g. a session key encrypted using a master key). Disadvantages of ECB include repetitions in the

message which can be reflected in cipher text, particularly when input data is a graphics file there can be several repetitions. Another weakness is due to encrypted message blocks being independent – this can make a brute force attack easier.

2.3.1.2.2.2 Cipher Block Chaining (CBC) Mode

In this mode, the message is broken into blocks, but these blocks are linked together in the encryption operation, i.e. cipher blocks are chained with plaintext, hence the name. This uses a known Initial Vector (IV) to start the process. Functionally, it can be described as:-

$$C_i = DES_{K1}(P_i \text{ xor } C_{i-1})$$

$$C_{-1} = \text{initial vector}$$

where P_i is the *ith* block of Plaintext

C_i is the *ith* block of Ciphertext

To overcome the problems of repetitions and order independence in ECB, it is desirable to have some way of making the ciphertext dependent on all blocks before it. This is what CBC gives us, by combining the previous ciphertext block with the current message block before encrypting. To start the process, an Initial Value (IV) is used, which is usually well-known (often all 0's), or is ECB encrypted, just before starting CBC use. CBC mode is applicable whenever large amounts of data need to be sent securely, provided that it is available in advance (e.g., email, FTP, web etc). In CBC mode, each ciphertext block is dependent on all message blocks before it.

CBC is the most common mode of use when data is available in advance. In CBC, a change in the message affects all the ciphertext blocks after the change as well as the changed block itself. The Initial Value (IV) must be known to both sender and the receiver. However, if IV is sent in the clear, an attacker can change bits of the first block, and change IV to compensate. Therefore, IV must be either a fixed value or it must be sent encrypted in ECB mode before rest of message.

In both ECB and CBC modes, one issue is how to handle the last block, which may well not be complete. In general, this block needs to be padded (typically with 0's), and then the padding must also be recognized at other end – it may be obvious (e.g. in plain text the 0 value should usually not occur), or otherwise an explicit last byte can be used as a count of how much padding was used (including the count itself).

2.3.1.2.2.3 Cipher Feedback (CFB) Mode

If the data is only available a bit (or byte) at a time (eg. terminal session, sensor value etc), then some other approach to encrypting must be used, so as not to delay the transmission of encrypted data. The idea here is to use the block cipher essentially as a *pseudo-random number* generator and to combine these *random* bits with the message. It should be noted that *XOR* is an easily inverted operator (can be undone just by performing *XOR* with the same value again). Again an IV is used at the start to get things going, then the ciphertext is used as the next input. As originally defined, the idea is to *consume* as much of the *random* output as needed for each message unit (bit/byte) before *bumping* bits out of the buffer and re-encrypting. This is wasteful though, and slows down the encryption process as more encryptions are needed for the same size of input data. An alternate way to think of it is to generate a block of *random* bits, consume them as message bits/bytes arrive, and when they are used up, only then feed a full block of ciphertext back. This is CFB-64 mode, the most efficient. This is the usual choice for stream-oriented data, and for authentication use.

In this mode, the message is treated as a stream of bits which are added (modulo 2) to the output of the block cipher. The result is feedback for next stage (hence the name). The CFB mode standard allows any number of bits to be used as feedback, e.g. CFB-1 means 1 bit is used as feedback, similarly CFB-4, CFB-8 etc. mean 4 or 8 bits are used as feedback, respectively. The CFB mode encryption can be described by the following function:-

$$C_i = P_i \text{ xor } DES_{K1}(C_{i-1})$$

$$C_{-1} = IV$$

Advantages and Limitations of CFB Mode

The following are some advantages and limitations of CFB mode:-

a. appropriate when data inherently arrives in bits/bytes
b. most common stream mode
c. a limitation is the need to stall while a block encryption is done after every 64 bits
d. block cipher is used in encryption mode at both ends
e. errors propagate for several blocks after the error

CFB is the usual stream mode. A possible problem with this mode is that if it is used over a *noisy* link, then any corrupted bit will destroy values in the current and next blocks (since the current block feeds as input to create the random bits for the next). So either this mode is used over a reliable network transport layer or OFB mode is used.

2.3.1.2.2.4 Output Feedback (OFB) Mode

The alternative to CFB is OFB. Here the generation of the *random* bits is independent of the message being encrypted (Figure 2.22). The advantage of this mode is that firstly, the random bits can be computed in advance. Secondly, any bit error only affects a single bit (unlike CFB mode, the subsequent bits are not affected). Thus, this mode is good for noisy links (e.g. satellite TV transmissions etc.)

In this mode, the message is treated as a stream of bits – output of block cipher is added to the message. This can be functionally described as below: -

$$C_i = P_i \text{ XOR } O_i$$
$$O_i = DES_{K1}(O_{i-1})$$
$$O_{-1} = IV$$

Advantages and Limitations of OFB Mode

Here are some advantages and limitations of the OFB mode:-

a. used where the error feedback is a problem, or where the encryptions must be done before the message is available
b. superficially similar to CFB, but the feedback is from the output of the block cipher and is independent of the message
c. a variation of the Vernam cipher, hence must never reuse the same sequence (key+IV)
d. sender and receiver must remain in sync, and some recovery method is needed to ensure that this happens
e. although originally specified with m-bit feedback in the standards, subsequent research has shown that only 64-bit OFB should ever be used

Because the random bits are independent of the message, they must never ever be used more than once (otherwise the two ciphertexts can be combined, canceling these bits, and leaving a book cipher to solve).

2.3.1.3Advanced Encryption Standard (AES)

2.3.1.3.1Overview

In September 1997, US NIST announced a call for candidate ciphers for its new Advanced Encryption Standard (AES), because clearly a replacement for DES was needed at that time [51]. The candidate ciphers were to be submitted by June 1998, and a finalist was selected in October 2000. In total 15 candidates were accepted in June 98 (6 were rejected as incomplete), and 5 were short-listed in August

99. Finally, Rijndael was selected as the AES finalist in October 2000. NIST has released all submissions and unclassified analyses.

The AES candidates are the latest generation of block ciphers, and have a significant increase in the block size - from the old standard of 64-bits up to 128-bits; and keys from 128 to 256-bits. In part this has been driven by the public demonstrations of exhaustive key searches of DES & RC-5 (at 64-bits).

2.3.1.3.2AES Requirements

The following requirements were specified by US NIST for AES submissions [51]:-

a. should be a private key symmetric block cipher
b. block size should be 128-bits, with 128/192/256-bit key size
c. should be stronger and faster than Triple-DES
d. active life should be between 20 to 30 years (with archival use beyond that)
e. should have full specification and design details
f. should have both C and Java implementations

2.3.1.3.3AES Shortlist

After exhaustive testing and evaluation, the following AES candidates [52] were short-listed in August 1999: -

a. MARS – a complex, fast, high security algorithm
b. RC6 – a very simple, very fast, low security algorithm
c. Rijndael – a clean, fast, good security algorithm
d. Serpent – a slow, clean, very high security algorithm
e. Twofish – a complex, very fast, high security algorithm

Further analysis of algorithms was done on the following basis:-

a. whether the algo has few complex rounds, or many simple rounds
b. whether it has refined existing ciphers, or is a brand new cipher

2.3.1.3.4AES Finalist – Rijndael

After exhaustive analysis and evaluation rounds, Rijndael – designed by Rijmen & Daemen in Belgium [53] – was selected as AES finalist [52]. It has the following attributes:-

a. 128-bit block size
b. 128, 192, or 256 bit key size

c. an iterative rather than a Feistel cipher (like IDEA)
d. treats data as 4 groups of 4 bytes
e. all operations can be combined into xor and table lookups – hence implementation can be very fast and efficient
f. has 9, 11, or 13 rounds, where each round consists of:-

- a byte substitution step (1 S-box used on every byte)
- a shift rows step (shuffle the bytes between groups)
- a mix columns step (matrix multiplication of groups with each other)
- an add round key step

Figure 2.24 shows a complete round of the AES algorithm.

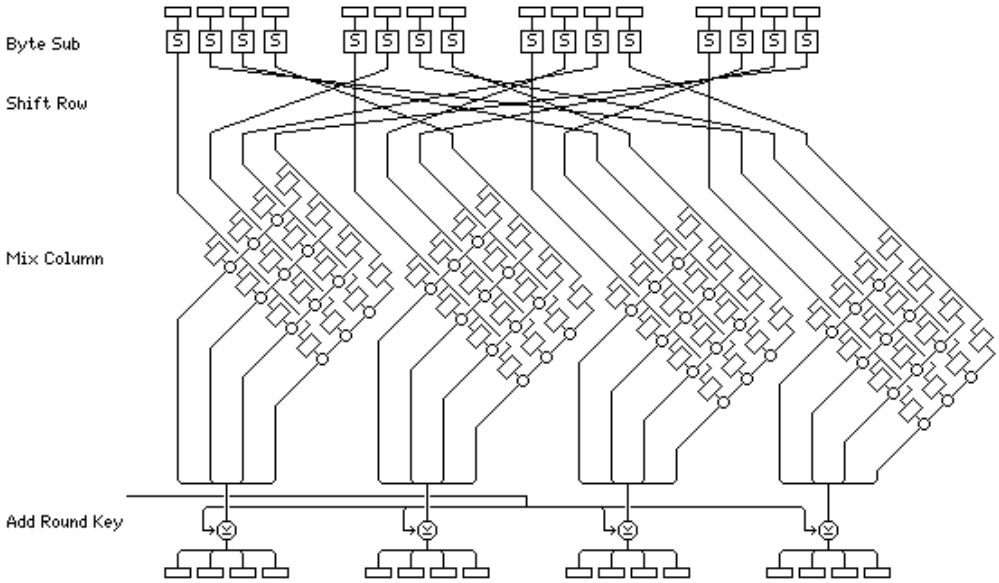


Figure 2.24: A complete round of the AES (Rijndael)

AES is now the new U.S. Advanced Encryption Standard [52]. The algorithm is oriented towards bytes (8 bits), but there is also emphasis on what the AES specification calls *words*, which are arrays of 4 bytes (or 32 bits, the size of an integer in Java).

The main mathematical difficulty with the algorithm is that it uses arithmetic over the field GF (2^8). Even the field itself only poses real difficulties with multiplication of field elements and with multiplicative inverses. Otherwise, the AES algorithm is just an annoying amount of details to orchestrate, but not really

difficult to implement. A more efficient implementation of multiplication in GF (2^8) requires methods that use tables to perform this operation.

Conventional block ciphers are always ugly, complicated, inelegant brutes, and the AES is no exception. In order to create such a cryptosystem, one must remember that anything done by encryption must be undone during decryption, using the same key since it is a conventional (symmetric key) system. Thus the focus is on various invertible operations. One standard technique in using the key is to derive a string somehow from the key, and use xor to combine it with the emerging ciphertext. Later, the same xor reverses this. Otherwise, there are mixing operations that move data around, and translation (or substitution) operations that replace one piece of data with another. This last operation is usually carried out on small portions of ciphertext using so-called S-boxes, which define replacement strings. One set of mixing, replacements, and exclusive-or with a string derived from the key is called a *round*. Then there will typically be a number of rounds. The AES uses different numbers of rounds for the different key sizes according to the Table 2.1 below. Originally the AES was going to support different block sizes, but they settled on just one size.

Key Sizes versus Rounds			
	Key Block Size *(Nk words)*	Plaintext Block Size *(Nb words)*	Number of Rounds *(Nr)*
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Table 2.1: AES key sizes vs. rounds

It may be noticed that a *word* is 4 bytes or 32 bits. The names *Nk, Nb,* and *Nr* are standard for the AES.

The particular form of this type of algorithm, with its rounds of mixing, substitution and exclusive-or with the key, was introduced with the official release of the Data Encryption Standard (DES) in 1977 and with work preceding the release. The DES has a block size of 64 bits and a very small key size of 56 bits. From the beginning, the key size of the DES was controversial, having been reduced at the last minute from 64 bits. This size seemed at the edge of what the National Security Agency (but no one else) could crack.

Now it is very easy to break, and completely insecure. The AES, with its minimum of 128 bits for a key should not be breakable by brute force attacks for a very long time, even with great advances in computer hardware.

2.3.1.3.5 Outline of the AES Algorithm.

Here is the AES algorithm in outline form, using Java syntax for the pseudo-code, and much of the AES standard notation (Figure 2.25):-

```
Constants: int Nb = 4;
          int Nr = 10, 12, or 14; // rounds, for Nk = 4, 6, or 8
Inputs: array in  of 4*Nb bytes // input plaintext
        array out of 4*Nb bytes // output ciphertext
        array w of 4*Nb*(Nr+1) bytes // expanded key
Internal work array:
       state, 2-dim array of 4*Nb bytes, 4 rows and Nb columns
Algorithm:
            void Cipher(byte[] in, byte[] out, byte[] w) {
              byte[][] state = new byte[4][Nb];
              state = in; // actual component-wise copy
              AddRoundKey(state, w, 0, Nb - 1);
              for (int round = 1; round < Nr; round++) {
                SubBytes(state);
                ShiftRows(state);
                MixColumns(state);
                AddRoundKey(state, w,
                    round*Nb, (round+1)*Nb - 1);
              }

            SubBytes(state);
              ShiftRows(state);
              AddRoundKey(state, w, Nr*Nb, (Nr+1)*Nb - 1);
              out = state; // component-wise copy
}
```

Figure 2.25: AES algorithm

Main points of the algorithm are as described below:

a. The multiplication operations in the algorithm are in GF(256) – this is not explicitly mentioned here, but the individual functions use it frequently.
b. *Nb* is a constant (= 4) that represents the number of *32-bit words* that make up a block for encryption.
c. *Nr* is the number of *rounds* or main iterations of the algorithm. The possible values depend on the three different possible key sizes.
d. *in* is the input block of 128 bits of plaintext, arranged as *4*Nb = 16 bytes*, numbered 0 to 15 and arranged in sequence.
e. *out* is the output block of 128 bits of ciphertext, arranged in the same way as *in*.
f. *state* is an internal array that is worked on by the AES algorithm. It is arranged as a *4xNb* 2-dimensinal array of bytes (that is, *4x4*).
g. *w* is the expanded key. The initial key is of size *4xNk bytes*, and this is expanded to the array *w* of *4*Nb*(Nr+1) = 16*(Nr+1) bytes* for input to the encryption algorithm. Each round uses *4*Nb bytes*, and each portion of *w* is used only once.
h. *SubBytes(state)* is a function that takes each byte of the state and independently looks it up in an *S-box table* to substitute a different byte for it. (The same S-box table is also used in the key expansion).
i. *ShiftRows(state)* is a function that simply moves around the rows of the state array.
j. *MixColumns(state)* is a function that does a much more complicated mix of the columns of the state array.
k. *AddRoundKey(state, w, param1, param2)* is a function that takes the *4*Nb*(Nr+1) bytes* of the expanded key, *w*, and does an exclusive-or of successive portions of the expanded key with the changing state array. The integer values *param1* and *param2* take on different values during execution of the algorithm, and they give the inclusive range of columns of the expanded key that are used.

2.3.1.4 Blowfish

Most of the encryption algorithms today are unavailable to the public – many of them are protected by patents (e.g. Khufu, REDOC II, and IDEA), or being kept secret by the governments (e.g. Skipjack and Capstone are protected by the U.S. government). Many of the other algorithms are available only in part (e.g. RC2, RC4, and GOST). *Bruce Schneier* – one of the world's leading cryptologists, and the president of *Counterpane Systems*, a consulting firm specializing in cryptography and computer security – designed the Blowfish algorithm and made it available in the public domain. Blowfish is a variable length key, 64-

bit block cipher. It was his intent from the outset of creating this new encryption algorithm to provide the world with a new encryption standard. The Blowfish algorithm was first introduce in 1993, and has not been cracked yet. It is also noteworthy to point out that this algorithm can be optimized in hardware applications, although it, like most other ciphers, is often used in software applications.

2.3.1.4.1Blowfish Design

There are two parts of this algorithm – the first part handles the expansion of the key and the second part performs encryption of the data.

Key Expansion

The first step in the algorithm is to break the original key into a set of subkeys. Specifically, a key of no more than *448* bits is separated into *4168* bytes. There is a *P-array* and four *32-bit S-boxes*. The P-array contains *18 32-bit* subkeys, while each *S-box* contains *256 32-bit* entries.

The following steps are used to calculate the subkeys:-

a. Initialize the *P-array* and *S-boxes*
b. xor *P-array* with the key bits. For example, *P1 xor (first 32 bits of key), P2 xor (second 32 bits of key), ...*
c. Use the above method to encrypt the all-zero string
d. This new output is now *P1* and *P2*
e. Encrypt the new *P1* and *P2* with the modified subkeys
f. This new output is now *P3* and *P4*
g. Repeat *521* times in order to calculate new subkeys for the *P-array* and the four *S-boxes*

Figure 2.26 shows a flowchart for the Blowfish algorithm. In the figure, the 64-bit input is denoted by *x*, while left and right halves of x are denoted by *xL* and *xR*. P_i denotes the *P-array* where *i* is the iteration number. Figure 2.27 shows working of the function *F*.

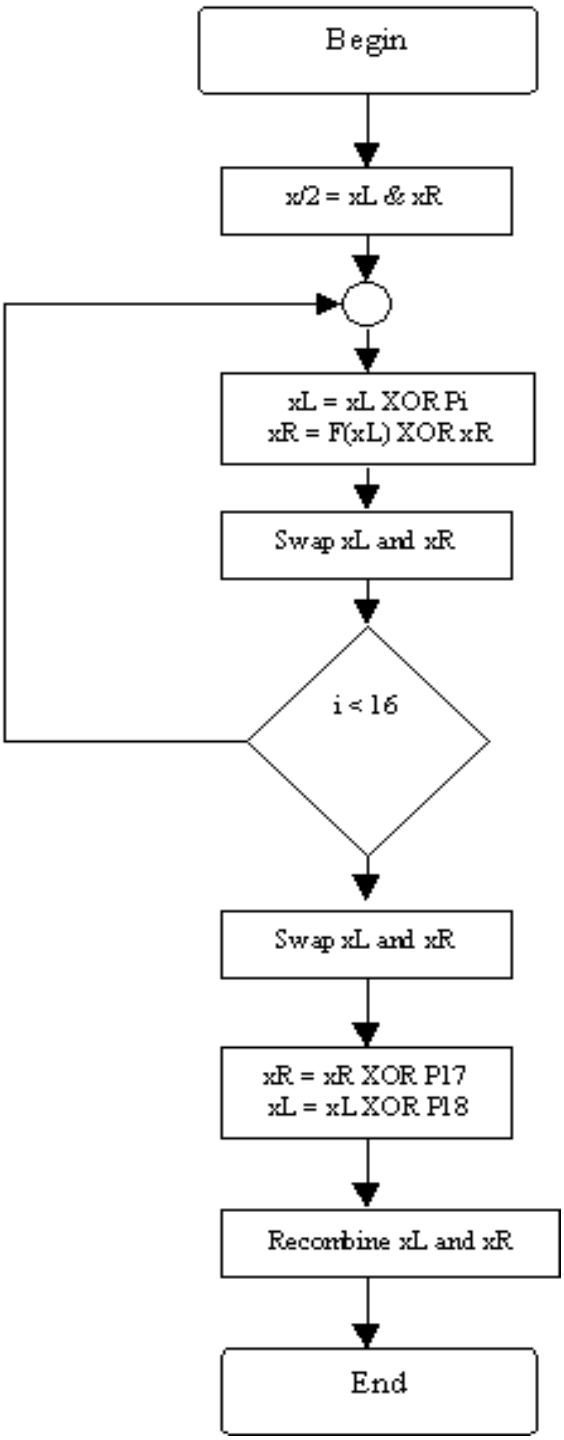


Figure 2.26: Flowchart for Blowfish algorithm

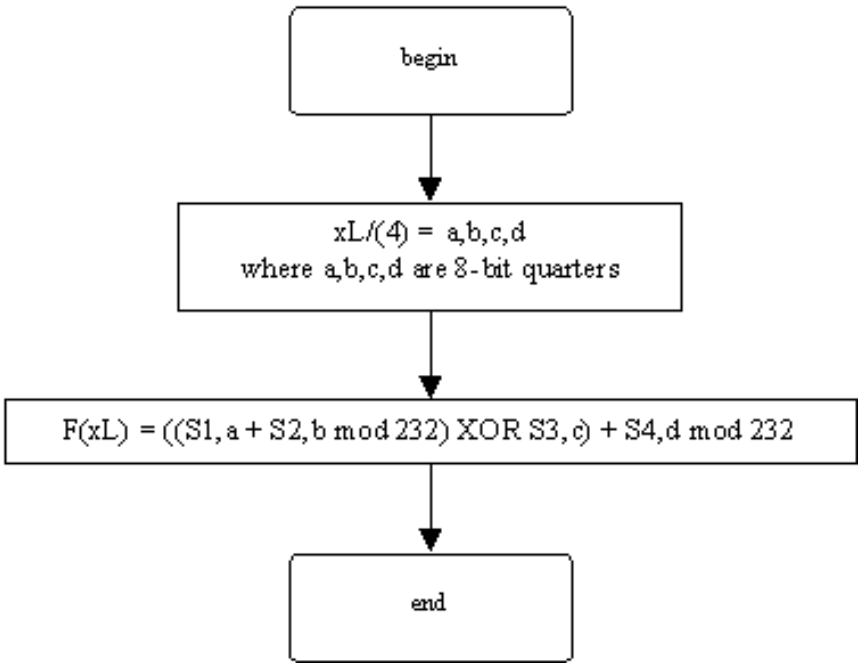


Figure 2.27: The function F of the Blowfish encryption

2.3.1.4.2Discussion

Implementing the Blowfish algorithm in a design course seemed like a viable option for encryption, considering that it was intended to be fast, compact, simple, and variably secure. Nevertheless, it should be pointed out that this algorithm is better suited to software than hardware. Specifically, the hardware implementation of the algorithm requires many states. In the state machine that the *eSAFE group* implemented, the master component has 10 states, while the encryption component has 18 states. An example of the size constraints in hardware can be seen by noting that for a *20000* gate chip, an optimized *VHDL* implementation of the Blowfish algorithm (using Altera's recommended state machine) requires *634* of *1152* cells (or *55%*) of the chip. This may not be of much concern if a larger chip is available or a simple method of transmitting data is used.

Schneier has encouraged the cryptanalysis of his blowfish algorithm and while it has been broken in part, it has not been broken significantly. Through cryptanalysis, it has been noted that weak keys can be detected. This attack does not work after 16 rounds of blowfish, however. Thus, it seems that the Blowfish is a reasonable encryption algorithm, especially for an unpatented one.

2.3.2 PUBLIC KEY ENCRYPTION

Traditional private key, secret key, or symmetric cryptography uses only one key for both encryption and decryption. The key is shared by both sender and receiver. If this key is disclosed, the communications are compromised. This type of cryptography is called symmetric because both parties involved (sender and receiver) are equal, in the sense that both can encrypt and decrypt messages using the same key. Hence, private key encryption does not protect sender from receiver forging a message and claiming that it was sent by sender.

So far all the cryptosystems discussed in this thesis, were private key (symmetric) systems. All classical and modern block and stream ciphers are of this form.
Public key or asymmetric cryptography involves the use of two keys:-

a. a *public-key*, which may be known to anybody, and can be used to encrypt messages, and verify signatures
b. a *private-key*, known only to the recipient, used to decrypt messages, and sign (or create) signatures

It is called asymmetric because the parties involved (sender and receiver) are not equal – those who encrypt messages or verify signatures cannot decrypt messages or create signatures. This is achieved through the clever application of number theoretic concepts. Development of public key cryptosystems is the single most significant advancement in the 3000-year history of cryptography.

Anyone knowing the public key can encrypt messages or verify signatures, but cannot decrypt messages or create signatures, though it may not seem intuitive. It works by the clever use of number theory problems that are easy one way but hard the other. Note that public key schemes are neither more secure than private key (security depends on the key size for both), nor do they replace private key schemes (they are too slow to do so), rather they complement them.

Note here the keys are created by the receiver not the sender (as is more common in private key systems, though either party can do so). In public key systems, the owner of the keys is the receiver, since they need to decrypt the message.

2.3.2.1 Theory of Public Key Cryptography

The public-key is easily computable from the private key and some other information about the cipher (a polynomial time (P-time) problem). However, knowing the public-key and public description of the cipher, it is still computationally infeasible to compute the private key (an NP-time problem). Thus the public-key may be distributed to anyone wishing to communicate securely with its owner. Although the secure distribution of the public-key is a non-trivial problem – the *key distribution* problem. Public key schemes utilize problems that are easy (P type) one way but hard (NP type) the other way, e.g. exponentiation vs. logarithms, multiplication vs. factoring. Consider the following analogy using padlocked boxes: traditional schemes involve the sender putting a message in a box and locking it, sending that to the receiver, and somehow securely also sending them the key to unlock the box. The radical advance in public key schemes was to turn this around, the receiver sends an unlocked box to the sender, who puts the message in the box and locks it (easy - and having locked it cannot get at the message), and sends the locked box to the receiver who can unlock it (also easy), having the key. An attacker would have to pick the lock on the box (hard). There is one problem - how does the sender know they have the correct unlocked box for the person they wish to send a message to? That is the key distribution problem [34].

2.3.2.2 Classes of Public-Key Algorithms

Public-Key Distribution Schemes (PKDS)

Public Key Distribution Schemes are used to securely exchange a single piece of information, usually a session key for a private-key scheme.

Public Key Encryption (PKE)

Public Key Encryption is used to encrypt any arbitrary message – anyone can use the public-key to encrypt a message, however, the owner uses their private-key to decrypt the messages. Any public-key encryption scheme can be used as a PKDS by using the session key as the message. Many public-key encryption schemes are also signature schemes (provided encryption and decryption can be done in either order).

Signature Schemes

Signature schemes are used to create a digital signature for some message. The owner uses private-key to sign (or create) the signature. Anyone can, then, use the public-key to verify the signature.

2.3.2.3Security of Public Key Schemes

The security of a public key scheme relies on a large enough difference in difficulty between the easy problem (encryption/decryption) and the hard problem (cryptanalysis).

As with private key schemes a brute force exhaustive search attack is always theoretically possible, but in practice the keys used are much too large (512 bits or larger) for this kind of attack. More generally, the hard problem is known, it's just made too hard to do in practice – it requires the use of very large numbers (>512 bits) which means that implementations are slow compared to private key schemes.

Public key schemes are no more or less secure than private key schemes – in both cases the size of the key determines the security. Note also that the key sizes cannot be compared – a 64-bit private key scheme has roughly similar security to a 512-bit RSA – both could be broken given sufficient resources. But with public key schemes at least there is usually a firmer theoretical basis for determining the security since it is based on well-known and well-studied number theory problems.

2.3.2.4 Public-Key Encryption – RSA

RSA (Rivest, Shamir, Adleman) public key encryption scheme is the best known and widely regarded as most practical public-key scheme, and was first proposed by Rivest, Shamir & Adleman (RSA) in 1977 [54]. It is based on exponentiation in a finite (Galois) field over integers modulo a prime. The security of RSA algorithm relies on the difficulty of calculating factors of large numbers:-

Exponentiation takes $O((\log n)^3)$ operations (an easy problem)

Factorization takes $O(e^{\log n \log \log n})$ operations (a hard problem)

RSA algorithm is patented in North America, and some other countries, which is also a source of legal difficulties in using this scheme.

2.3.2.4.1RSA Setup

Initially, each user generates their public/private key pair in the following manner:-

a. select two large prime numbers at random (each of approximately100 digits), p, q
b. compute their system modulus $N=p.q$
c. select at random the encryption key e, where $e<N$, $\gcd(e,\phi(N))=1$
d. solve the following congruence to find the decryption key d:-

$e.d \equiv 1 \bmod \phi(N)$ and $0<=d<=N$

i. make public their public encryption key: $K=\{e,N\}$

ii. keep secret their private decryption key: $K^{-1}=\{d,N\}$

This key setup is done only once when a user establishes (or replaces) their public key. The exponent *e* is usually fairly small – it just needs to be relatively prime to ø(N). The decryption exponent *d is computed as inverse of e*. It is critically important that the private key {d,N} is kept secret, since if it becomes known, the system can be broken.

2.3.2.4.2RSA Parameter Selection

The parameters p and q need to be suitably large prime numbers. Usually the encryption exponent e is chosen to be a small number which must be relatively prime to ø (N). Typically, e may be the same for all users (provided certain precautions are taken). Originally a value of 3 was suggested which is now regarded as too small, and usually the value $2^{16}-1 = 65535$ is often used. It should be noted that the decryption exponent d would be large if e is small.

2.3.2.4.3Theory behind RSA

RSA works because of Euler's Generalization of Fermat's Theorem:-

if $N = pq$ where p, q are primes

then $X^{\phi(N)} = 1 \bmod N$

for all X not divisible by p or q, ie $\gcd(X,\phi(N))=1$
where ø(N)=(p-1)(q-1)

2.3.2.4.4RSA Security

The security of RSA rests on the difficulty of computing ø(N) from e and N, which is assumed to require factoring of N. Assuming that 1e+14 operations is regarded as a limit for computational feasibility (this is about 3 MIPS-years – a MIPS-year is 3e+13 operations/year), Table 2.2 shows MIPS-years required using various algorithms, with different sizes of N. Barring a dramatic breakthrough 1024+ bit RSA looks secure for now.

Decimal Digits	Bits	When	MIPS-Years	Algo Used
100	332	Apr 91	7	Quadratic Sieve
110	365	Apr 92	75	Quadratic Sieve
120	398	Jun 93	830	Quadratic Sieve
129	428	Apr 94	5000	Quadratic Sieve
130	431	Apr 96	500	Number Field Sieve
140	464	Feb 99	1500	Number Field Sieve

Table 2.2: MIPS-Years required for various sizes of N

2.3.3 Performance of Encryption Algorithms

The design goal of any encryption algorithm is security against undesireable attacks. However, in the real world, performance and implementation costs are also important concerns. As regards performance of the algorithms, a lot more factors such as complexity of a round, and the number of rounds may also be considered.

In [63], the implemented algorithms are DES, 3DES, AES (Rijndael), Blowfish, RSA and their performance has been compared by encrypting input files of varying contents and sizes. The algorithms have been implemented in a uniform language, using their standard specifications, and tested on two different hardware platforms, to compare their performance. In the end, the results were presented which conclude that the Blowfish is the fastest algorithm. Though security was not catered for, in practice, however, one would consider the security first.

We selected secret key encryption algorithms as they are more common and public key encryption algorithms are mainly used for secret key exchange between two parties. Secret key algorithms are 10 to 100 times faster than public key algorithms. We will use blowfish algorithm, as in [63] it is proved that blowfish is the fastest algorithm among all other secret key encryption algorithms.

Chapter 3
Implementation of Image Steganographic System

3.1 Introduction

Every time researchers have very limited parameters to play within when proposing a new algorithm. In image steganography the main parameters to focus on are:-

a. Robustness
b. Security
c. Capacity

The problem in image steganography is to minimize the chances that the presence of hidden/secret message is identified. There is a tradeoff between the making image imperceptible and the embedding capacity of the stego-image, for a given cover image. By increasing the capacity, image quality is degraded and invisibility is not achieved. By maximizing imperceptibility, the embedding capacity is reduced.

This chapter presents the design and implementation details of the proposed algorithms.

3.2 Proposed System

This thesis work presents an image steganographic system which contains three (03) devised algorithms to improve the image imperceptibility. Algorithms are as follows:-

a. Image Steganography Technique for Colored Images using Wavelet Transform
b. Image Steganography Technique for Colored Images using Contourlet Transform
c. Image Steganography Technique for Colored Images using Huffman Encoding with Symlet Wavelet Transform

3.2.1 ALGO A: Image Steganography Technique for Colored Images using Wavelet Transform

Algorithm has been implemented in MATLAB 7.9 and Visual Studio 2010. Devised method consists of two main processes. First one deals with the MATLAB implementation which passes some controls to

Visual Studio 2010 for C# implementation of blowfish algorithm. The other process then returns back the control to MATLAB from Visual Studio 2010. Main steganographic routines have been developed in MATLAB using wavelet toolbox functions and Blowfish algorithm for enhanced security in C# using Visual Studio 2010. Following is the pseudo code:

Step 1: image histogram test
Input: Cover image
Output: Cover image
Action: If the cover image passes the histogram test
Then
accept the cover image
Else
search for another cover image.

Step 2: image corrections
Input: Cover image
Output: Processed cover image
Action:
For each pixel of cover image apply level correction
For each pixel of cover image apply contrast correction
For each pixel of cover image apply color balance correction

Step 3: Apply wavelet transform
Input: Processed cover image
Output: Transformed image
Action: Apply 2D wavelet transform

Step 4: Calculate threshold (Th)
Input: Transformed image
Output: number of pixels can be changed (n), DWT of the cover image
Action: Calculate Th
Th = (correlation factor/Number of coefficient) $\sum$ (mod value of all 2D wavelet coefficients)
Select DWT coefficient of every pixel in cover image

if coefficient value < Th

then

keep coefficient index and n=n+1

Step 5: Message encryption process

Input: Information

Output: encrypted information

Action: Pass control from Matlab 7.9 to Visual Studio 2010

Encrypt information with Blowfish

Pass control back to Matlab

Step 6: Bit streaming

Input: n, encrypted information

Output: 1D bit streamed information

Action: convert the encrypted information to 1D bit stream using n

Step 7: 2D wavelet transform of encrypted information

Input: encrypted bit stream of the message

Output: DWT transform of the encrypted message.

Action: transform the encrypted bit stream of the message to Wavelet domain

Step 8: Coefficient placement

Input: 2D wavelet transform of cover image and 2D wavelet transform of the encrypted information.

Output: stego image

Action:

a. Put coefficients of wavelet transform of encrypted information in the indexes stored in Step 4

b. Perform inverse wavelet transform

Figure 3.1 illustrates flow chart of ALGO A.

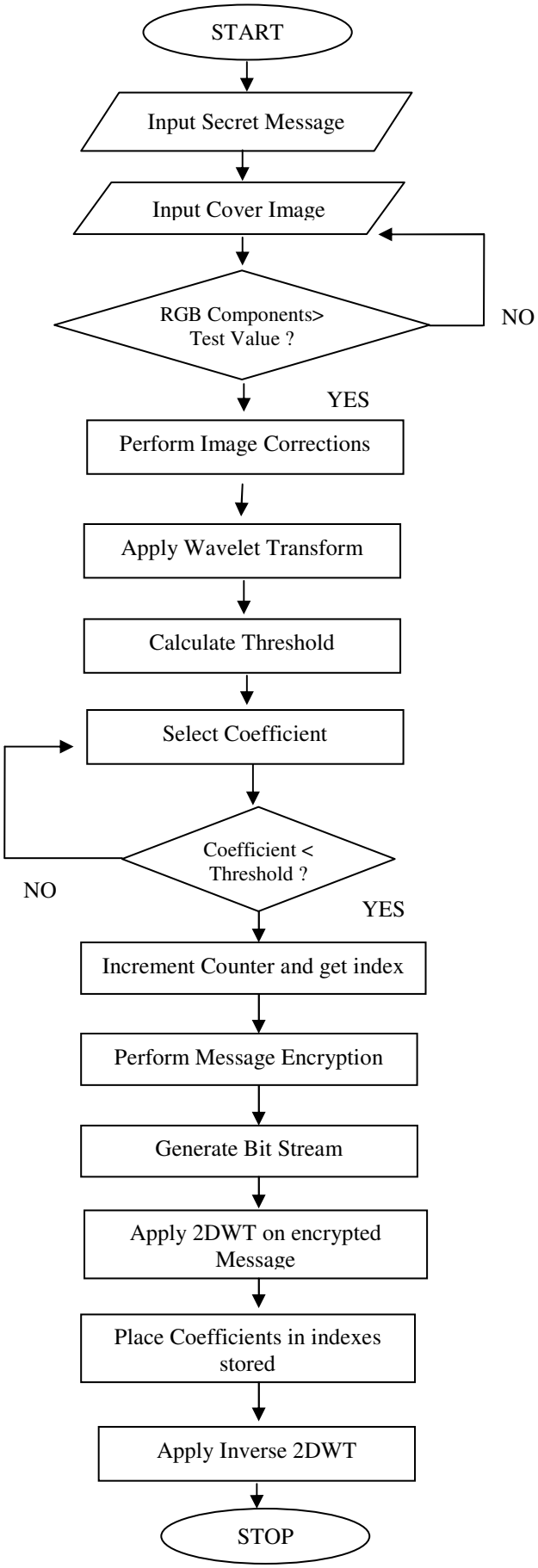


Figure 3.1: ALGO A

3.2.1.1ALGO A Implementation

This algorithm accepts the input cover image format and input information to be encoded. It then processes the *Function hist* on input cover image and returns 1 if test is passed and 0 otherwise. The next step is image preprocessing, which is performed using *Fucntion imagecorrection*. This function accepts the tested cover image and performs following three (03) preprocessing functions:

a. level correction
b. contrast correction
c. balance correction

The above three functions have been performed using image processing toolbox. The next step as also explained in flow chart is to get wavelet coefficients. This task is done by using *Function 2dwt*. In the proposed algorithm, Symlet wavelet has been used. It accepts the corrected image and performs wavelet functions using the wavelet toolbox.

The transformed cover image or coefficients are passed to *Function thresh* and it processes the inputs using the following equation:

Th = (correlation factor/Number of coefficient) $\sum$ (mod value of all 2D wavelet coefficients) … (1)

It returns the number of pixels that can be changed (n).

The second phase of the algorithm deals with the input message. The input message has been accepted using MATLAB 7.9. Now it is passed to Blowfish encryption algorithm using MEX file implemented in C Sharp or C#. The blowfish algorithms encrypts the message and sends the control back to MATLAB program with the processed input message which is now an encrypted one.

The encrypted message and the number of pixels (n) are now passed to *Function bit_stream* for converting the encrypted message to 1D bit stream using n. This bit stream is then used with the wavelets to get it transformed into wavelet domain using wavelet toolbox.

The transformed cover image and transformed message bit stream have been further processed. In this step, the coefficients of wavelet transform of encrypted message are placed in to the indexes stored earlier in ‘n’ of the transformed cover image. At end the inverse wavelet transform is performed.

3.2.2 ALGO B: Image Steganography Technique for Colored Images using Contourlet Transform

Algorithm has been implemented in MATLAB 7.9 and Visual Studio 2010. Devised method consists of two main processes. First one deals with the MATLAB implementation which passes some controls to Visual Studio 2010 for C# implementation of blowfish algorithm. The other process then returns back the control to MATLAB from Visual Studio 2010. Main steganographic routines have been developed in MATLAB and Blowfish algorithm for enhanced security in C# using Visual Studio 2010. Following is the pseudo code:

Step 1: image histogram test
Input: Cover image
Output: Cover image
Action: If the cover image passes the histogram test
    Then
        accept the cover image
    Else
        search for another cover image

Step 2: image corrections
Input: Cover image
Output: Processed cover image
Action:
For each pixel of cover image apply level correction
For each pixel of cover image apply contrast correction
For each pixel of cover image apply color balance correction

Step 3: Apply contourlet transform
Input: Processed cover image
Output: Transformed image
Action: Apply 2D contourlet transform

Step 4: Calculate threshold (Th)
Input: Transformed image
Output: number of pixels can be changed (n), contourlet transform of the cover image

Action: Calculate Th

Th = (correlation factor/Number of coefficient) ∑ (mod value of all 2D contourlet transform coefficients)

Select contourlet transform coefficient of every pixel in cover image

if coefficient value < Th

then

keep coefficient index and n=n+1

Step 5: Message encryption process

Input: Information

Output: encrypted information

Action: Pass control from Matlab 7.9 to Visual Studio 2010

Encrypt information with Blowfish

Pass control back to Matlab

Step 6: Bit streaming

Input: n, encrypted information

Output: 1D bit streamed information

Action: convert the encrypted information to 1D bit stream using n

Step 7: 2D contourlet transform of encrypted information

Input: encrypted bit stream of the message

Output: contourlet transform of the encrypted message.

Action: transform the encrypted bit stream of the message to contourlet domain

Step 8: Coefficient placement

Input: 2D contourlet transform of cover image and 2D contourlet transform of the encrypted information.

Output: stego image

Action:

a. Put coefficients of contourlet transform of encrypted information in the indexes stored in Step 4

b. Perform inverse contourlet transform

Figure 3.2 illustrates flow chart of ALGO B

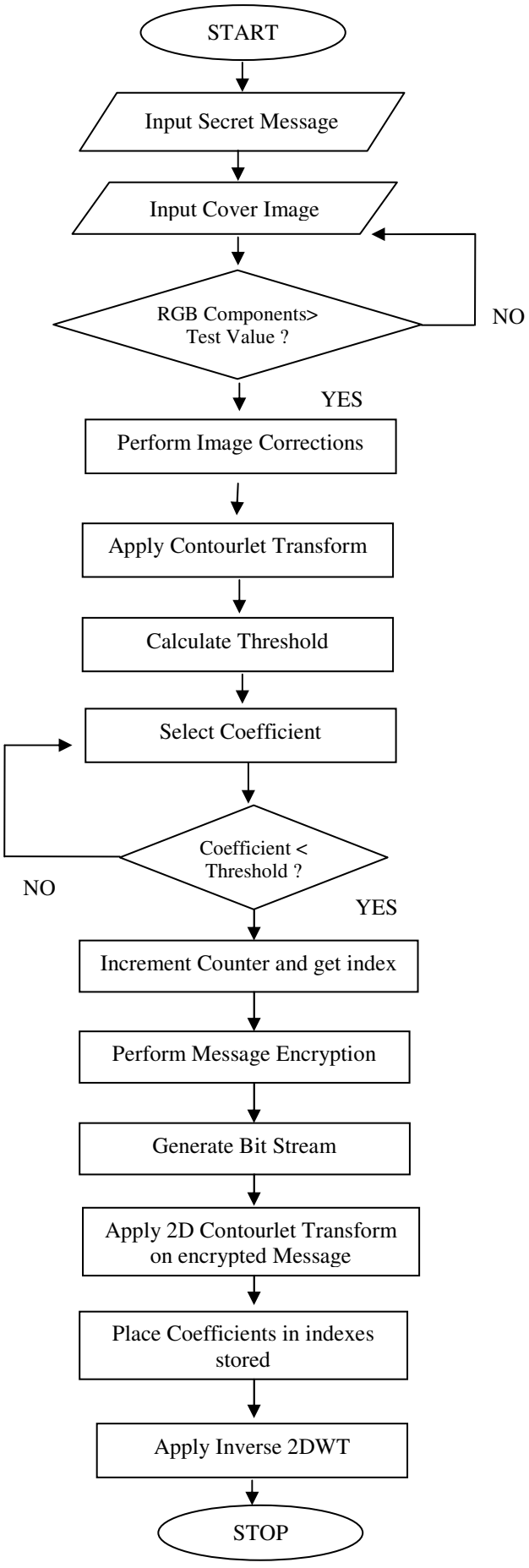


Figure 3.2: ALGO B

3.2.2.1ALGO B Implementation

This algorithm starts with an input cover image of JPEG format and input information to be encoded. It then processes the *Function hist* on input cover image and returns 1 if test is passed and 0 otherwise. The next step is image preprocessing, which is performed using *Fucntion imagecorrection*. This function accepts the tested cover image and performs following three (03) preprocessing functions:

a. level correction
b. contrast correction
c. balance correction

The above three functions have been performed using image processing toolbox. The next step as also explained in flow chart is to get contourlet coefficients. This task is done by using contourlet toolbox function for 2D contourlets. It accepts the corrected image and performs contourlet functions.

The transformed cover image or coefficients are passed to *Function thresh* and it processes the inputs using the following equation:

Th = (correlation factor/Number of coefficient) $\sum$ (mod value of all 2D wavelet coefficients) … (1)

It returns the number of pixels that can be changed (n).

The second phase of the algorithm deals with the input message. The input message has been accepted using MATLAB 7.9. Now it is passed to Blowfish encryption algorithm using MEX file implemented in C Sharp or C#. The blowfish algorithms encrypts the message and sends the control back to MATLAB program with the processed input message which is now an encrypted one.

The encrypted message and the number of pixels (n) are now passed to *Function bit_stream* for converting the encrypted message to 1D bit stream using n. This bit stream is then used with the contourlets to get it transformed into contourlet domain using contourlet toolbox.

The transformed cover image and transformed message bit stream have been further processed. In this step, the coefficients of contourlet transform of encrypted message are placed in to the indexes stored earlier in ‘n’ of the transformed cover image. At end the inverse of contourlet transform is performed.

3.2.3 ALGO C: Image Steganography Technique for Colored Images using Huffman Encoding with Symlet Wavelet Transform

The proposed algorithm was implemented in MATLAB 7.9. It is divided into two main phases. Following is the pseudo code of devised method:

Phase I: Huffman Encoding

Step I: Obtain secret Message

Step II: Perform Huffman encoding on the secret message.

Step III: Create Huffman Table.

Step IV: Perform the binary conversion.

Step V: Make groups of 3bits each

Check last group (if number of bits! = 3)

Add zero at end of bits (0)

Phase II:

Step I: Select colored Cover Image.

Step II: Separate Red, Green and Blue Components (RGB).

Step III: Apply 2DWT on each component separately.

Step IV: Embed Huffman codes (calculated in Phase I)

Step V: Add RGB Components

Step VI: Apply Inverse 2DWT

To embed Huffman codes, firstly divide Huffman groups by 3. For example 12 groups are formed by Huffman encryption phase. By dividing it by three (03), 3 sets are obtained of 4 groups each. Now select 3 consecutive bits of first set and insert into the Least Significant Bit (LSB) of 2DWT coefficient of red component. It is done for the rest of three groups in set 1. It is repeated with set 2 for 2DWT coefficient of green component and set 3 for 2DWT coefficients of blue component.

Figure 3.3 (a and b) illustrates flow chart of ALGO C.

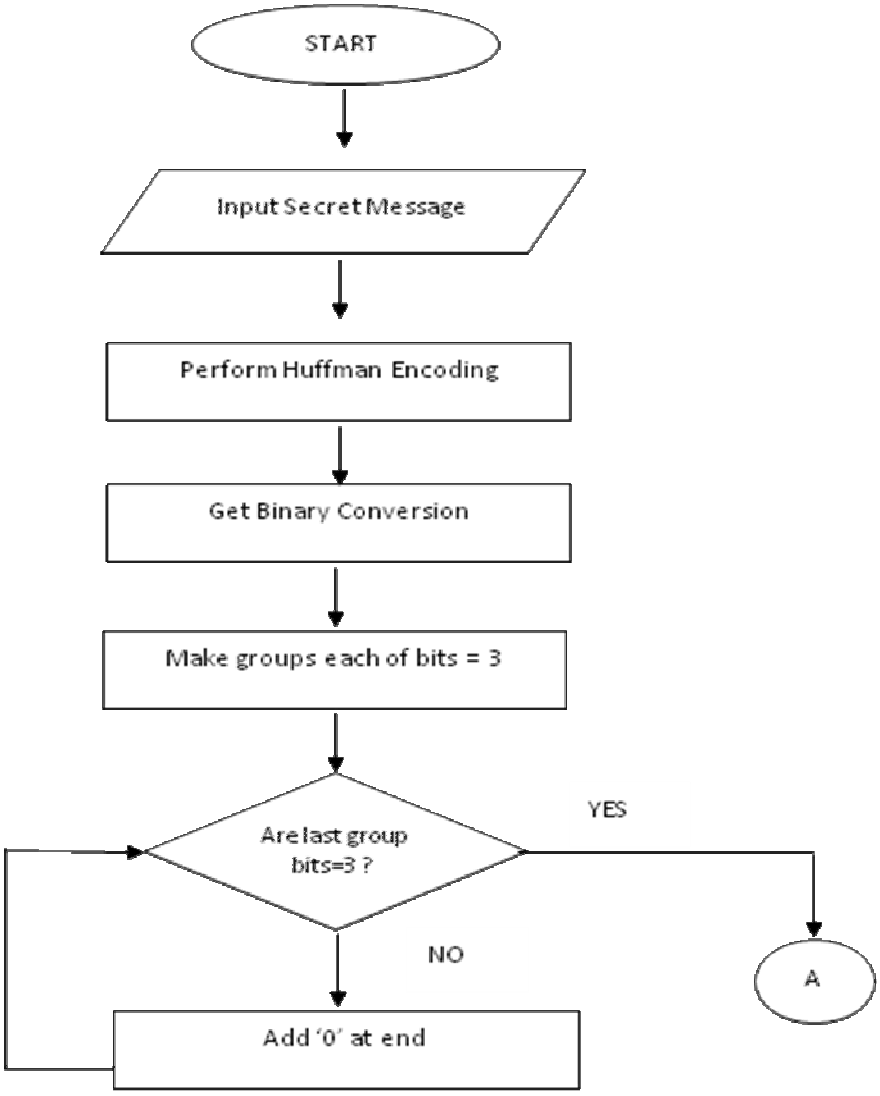


Figure 3.3 a: Huffman Encoding step of ALGO C

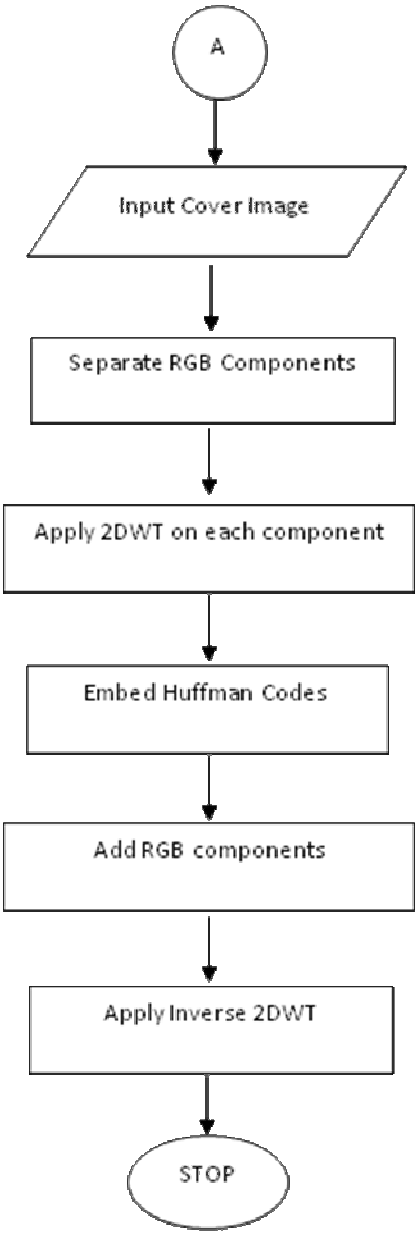


Figure 3.3 b: PHASE II of ALGO C

3.2.3.1 ALGO C Implementation

The algorithm starts with the input secret message in any Microsoft Word file (.doc/.docx) which is to be encoded and an input cover image of any size in JPEG format. This algorithm consists of two phases. In the first phase input message is processed. The input information is then passed to the *Function for huffman encoding*. This function comprises of many small sub-routines. The end result is to generate a Huffman Table. Each entry of Huffman table is converted to binary for code generation. The last step of this phase is to group all the binary bits generated of three (03) bits each. It has been ensured that last group also contains three bits like other groups. If the last group doesn't, we have to add bit '0' at end of last group and increment the value of flag set. This flag helps us to decode the information at reciever end.

The second phase of algorithm deals with the image processing and encoding. The input cover image has been tested first to check if it is a colored image or not. If it is, it has been proceeded to next step else it returns value '0'. The passed input cover image is then separated into Red, Green and Blue (RGB) components and stores each of the components in separate arrays.

Now, this algorithm employs wavelet transform. The 2D wavelet transform is performed on each component using wavelet toolbox. Symlet wavelets are used for transforming in wavelet domain. Now the resulted wavelet coefficients have been processed with Huffman codes/groups of bits. The total number of groups generated has been divided into three (03) main sets. The three consecutive bits of first set have been selected and inserted into the Least Significant Bit (LSB) of 2DWT of red component. It is done until all the groups in set 1 have been covered. The algorithm repeatedly inserts LSB of set 2 in the 2DWT coefficients of green component and set 3 in 2DWT coefficients of blue component. The final stego image has been formed by adding all the three modified RGB components in a single image. The last step of second phase or perhaps of algorithm is to get the inverse of wavelet transform on final stego image.

3.2.4 Blowfish Implementation

The blowfish algorithm has been implemented in C Sharp or C# using Visual Studio 2010 on Windows 7 Platform. It has been called by ALGO A and ALGO B using MEX file. Following is the detail of algorithm:-

The MATLAB code passed input message only to the Blowfish. It stores this information as its one of the inputs. The other input is a 128 bit Blowfish key which has been used for encryption. There are two options, either to load it from a file or to generate it. Using the enrption key, P-box has been generated in the form of an array consisting of 18 words, where each word is of 32 bits. The other S-box array has been generated by using the encryption key by repreated application of the Blowfish algorithm itself. The size of this array is 4x256 words, where each word is 32 bits long. There is another array used to store the Blowfish key. In this, each element is a 32 bit word and for a 128 bit key, the array has four elements.
To check whether encryption key has been loaded yet or not, a boolean variable has been used to keep track of it.

This algorithm has been programmed using Electronic Code Book (ECB) of Blowfish encryption. In this mode, the data from the input file is divided into data blocks of 64 bits each. Then each data block is encrypted and writes it to the output file for MATLAB code. The data block encryption has been done with the number of rounds.

3.2.5 Extraction Algorithms

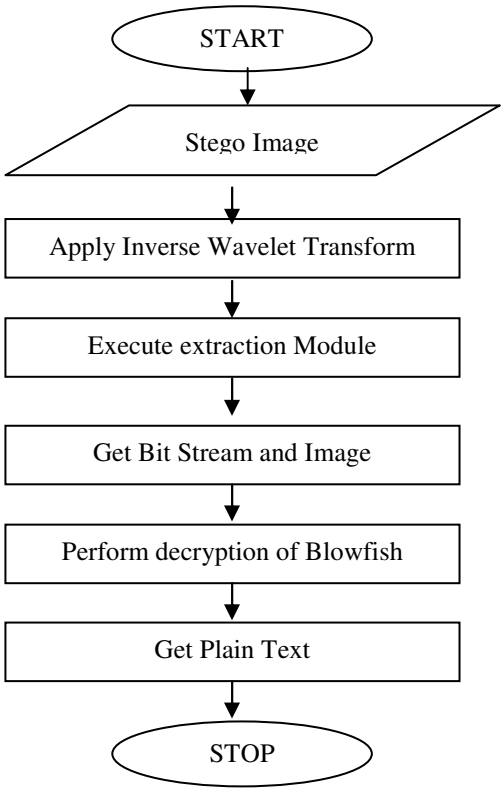


Figure 3.4 Extracting Message and Image from ALGO A

Figure 3.4 illustrates the extraction algorithm to extract the cover image and secret message. It first take the stego image and then employs inverse wavelet transform on it. After getting inverse of wavelet transform the processed data has been passed to extraction module. This module results in bit stream and cover image. The obtained bit stream is decrypted by using the decryption module of Blowfish to get input plain text that was encoded.

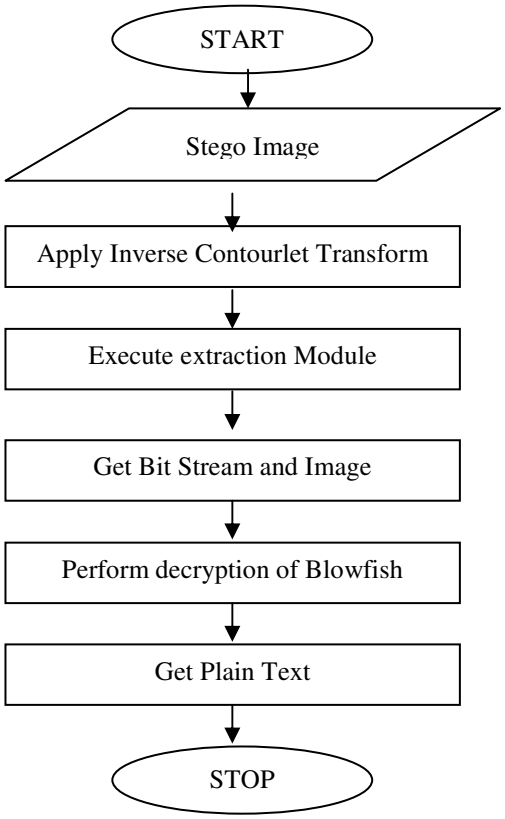


Figure 3.5 Extracting Message and Image from ALGO B

For extracting message and image from ALGO B, the Figure 3.5 explains the extraction algorithm. It starts with the stego image. To extract the message and input cover image from ALGO B the inverse of contourlet transform has been obtained and passed it to extraction module. This module further processes the extracted data and generates the bit stream and input cover image. The decryption of bit stream is done by using the blowfish decryption algorithm. The image and input secret message have been obtained as a result of ALGO B extraction.

Chapter 4
Results and Discussion

4.1 Introduction

In this section, the performance of three proposed algorithms has been measured by using following three sets of image sizes, each contains ten (10) colored images:

a. 256x256
b. 306x648
c. 512x512

The reason for selecting the above image sizes is that the performance has been improved against the work presented by Ali Al-Taby in [62] and in [62] the image size taken was 306x648. Other image sizes have also been collected to check the trend. These input test images used as cover images vary from complex to smooth ones.

Proposed algorithms are as follows:-

a. ALGO A: Image Steganography Technique for Colored Images using Wavelet Transform
b. ALGO B: Image Steganography Technique for Colored Images using Contourlet Transform
c. ALGO C: Image Steganography Technique for Colored Images using Huffman Encoding with Symlet Wavelet Transform

To show the distortion of the final stego image, Peak Signal to Noise Ratio (PSNR) and Mean Squared Error (MSE) is used with the payload. Payload is calculated as the total number of bits that can be embedded into the number of bits of input cover image. It varies with the smoothness of an image. The payload value is taken in percentage (%). The PSNR is actually the ratio of maximum possible power of signal and the power of noise that can affect the reliability of signal. It is expressed in logarithmic decibels because most of the signals have very wide dynamic range. The higher the value of PSNR, the better is the performance. It can easily be calculated using MSE which is defined for two i x j images as given in Equation (1) and (2):-

$$MSE = \frac{1}{MN} \sum_{i=0}^{M} \sum_{j=0}^{N} (Xij - \overline{Xij}) \quad (1)$$

$$PSNR\ (dB) = 10\log_{10} \left(\frac{I^2}{MSE}\right) \quad (2)$$

X_{ij} represents the i^{th} rows and j^{th} columns of original image and $\bar{X}_{ij}$ represents the i^{th} rows and j^{th} columns of transformed image. Higher the PSNR value means more difficult to perceive that any hidden message is hidden.

Correlation factor (α) is a statistical measure of interdependence between two variables. It measures the strength of straight line or linear relationship between two variables. It takes on values ranging from -1 to +1. Correlation factor is used to calculate the threshold value.

In the experiments performed, input text information is taken as user's message and embedded into each of the input cover images. The input information is in Microsoft Word file (.doc/.docx) and is processed by C # MEX file. This file consists of combination of characters, numeric values, and special characters.

4.2 System Specifications

To measure the performance of proposed algorithms the specification of system has been used are given in Table 4.1.

Algorithms	Processor	RAM	Hard Disk	Operating System
Image Steganography Technique for Colored Images using Wavelet Transform	2 GHz (core 2 Duo)	4 GB	224 GB	Windows 7
Image Steganography Technique for Colored Images using Contourlet Transform	2 GHz (core 2 Duo)	4 GB	224 GB	Windows 7
Image Steganography Technique for Colored Images using Huffman Encoding with Symlet Wavelet Transform	2 GHz (core 2 Duo)	4 GB	224GB	Windows 7

Table 4. 1 : System Specifications

4.3 Comparison of ALGO A and B

Ten experiments have been performed by varying the correlation factor and input cover images of three different sizes. In each experiment, Symlet 4 wavelet family has been used. All of these experiments

make use of horizontal, vertical and diagonal details and approximation details of level 3 decomposition. The details of these experiments are given experiment wise. Taking a single image for each experiment, both ALGOs A and B have been executed. Results have been tabulated to give a comparison of both.

4.3.1 Experiments

Experiment No. 1: Image Name PATTERN

		ALGO A: Wavelets			ALGO B: Contourlets		
Image Size	**Correlation Factor (α)**	**MSE (%)**	**Payload (%)**	**PSNR (dB)**	**MSE (%)**	**Payload (%)**	**PSNR (dB)**
256x256	0.1	0.32	30.06	50.67	0.27	30.06	52.17
	0.2	0.46	39.94	47.21	0.43	39.94	48.71
	0.3	0.61	45.07	45.02	0.58	45.07	46.52
	0.4	1.51	51.19	41.21	1.48	51.19	42.71
	0.5	1.84	55.5	37.67	1.81	55.5	39.17
	0.6	2.69	62.93	32.81	2.66	62.93	34.31
	0.7	3.66	64.72	29.97	3.63	64.72	31.47
	0.8	4.87	67.47	27.77	4.84	67.47	29.27
	0.9	6.13	69.21	24.32	6.1	69.21	25.82
	1	7.92	72.02	22.01	7.89	72.02	23.51
306x648	0.1	0.34	38.41	52.65	0.27	38.41	54.35
	0.2	0.55	42.71	51.18	0.48	42.71	52.88
	0.3	0.7	49.95	49.99	0.63	49.95	51.69
	0.4	1.26	56.87	45.3	1.19	56.87	47
	0.5	1.94	61.94	41.59	1.87	61.94	43.29
	0.6	2.79	65.86	38.56	2.72	65.86	40.26
	0.7	3.76	69.05	36.03	3.69	69.05	37.73
	0.8	4.95	71.63	33.83	4.88	71.63	35.53
	0.9	6.26	73.84	31.85	6.19	73.84	33.55
	1	7.92	74.82	28.94	7.85	74.82	30.64
512x512	0.1	0.37	42.36	55.63	0.28	42.36	57.53
	0.2	0.52	46.68	53.15	0.43	46.68	55.05
	0.3	0.76	51.07	51.13	0.67	51.07	53.03
	0.4	1.26	57.81	48.53	1.17	57.81	50.43
	0.5	1.97	63.37	44.16	1.88	63.37	46.06
	0.6	2.83	66.37	41.41	2.74	66.37	43.31
	0.7	3.86	71.59	38.63	3.77	71.59	40.53
	0.8	5.28	74.01	36.17	5.19	74.01	38.07
	0.9	6.51	76.89	34.72	6.42	76.89	36.62
	1	7.39	79.51	31.09	7.3	79.51	32.99

Table 4. 2: Image Name PATTERN

It has been observed from Table 4.2 that image name 'PATTERN' is used for simulating results and as the correlation factor keeps on increasing, the value of PSNR value deteriorates for each image size. For correlation factor 0.5 and image size 512x512, the ALGO A: Wavelets gives PSNR value equals to 44.16. ALGO B: Contourlets gives 46.06 as PSNR value for the same correlation factor and image size. So, the ALGO B has performed well as compared to the ALGO A in all the results.

Experiment No. 2: Image Name BIGCATS

		ALGO A: Wavelets			**ALGO B: Contourlets**		
Image Size	**Correlation Factor (α)**	**MSE (%)**	**Payload (%)**	**PSNR (dB)**	**MSE (%)**	**Payload (%)**	**PSNR (dB)**
256x256	0.1	0.31	30.04	49.63	0.26	30.04	51.13
	0.2	0.44	39.91	47.17	0.41	39.91	48.67
	0.3	0.58	45.04	44.97	0.55	45.04	46.47
	0.4	1.47	51.06	41.17	1.44	51.06	42.67
	0.5	1.83	55.3	37.64	1.8	55.3	39.14
	0.6	2.67	62.91	31.72	2.64	62.91	33.22
	0.7	3.64	64.69	29.92	3.61	64.69	31.42
	0.8	4.85	67.46	27.73	4.82	67.46	29.23
	0.9	6.11	69.19	24.28	6.08	69.19	25.78
	1	7.89	71.98	21.89	7.86	71.98	23.39
306x648	0.1	0.35	38.37	51.62	0.28	38.37	53.32
	0.2	0.52	42.68	49.15	0.45	42.68	50.85
	0.3	0.67	49.92	47.96	0.6	49.92	49.66
	0.4	1.23	56.84	44.27	1.16	56.84	45.97
	0.5	1.91	61.91	40.56	1.84	61.91	42.26
	0.6	2.76	65.83	38.53	2.69	65.83	40.23
	0.7	3.76	69.02	36	3.69	69.02	37.7
	0.8	4.92	71.6	32.8	4.85	71.6	34.5
	0.9	6.23	73.81	29.82	6.16	73.81	31.52
	1	7.89	74.79	27.91	7.82	74.79	29.61
512x512	0.1	0.39	42.38	54.65	0.3	42.38	56.55
	0.2	0.51	46.7	53.17	0.42	46.7	55.07
	0.3	0.78	51.09	51.15	0.69	51.09	53.05
	0.4	1.24	57.83	47.55	1.15	57.83	49.45
	0.5	1.99	63.39	43.18	1.9	63.39	45.08
	0.6	2.85	66.27	39.43	2.76	66.27	41.33
	0.7	3.88	71.61	37.65	3.79	71.61	39.55
	0.8	5.04	74.03	35.19	4.95	74.03	37.09
	0.9	6.36	76.91	33.74	6.27	76.91	35.64
	1	7.32	79.53	29.81	7.23	79.53	31.71

Table 4. 3: Image Name BIGCATS

It can be noted from all the results tabulated in Table 4.3 that ALGO B is better than ALGO A. Image 'BIGCATS' is used as test image and correlation factor varies from 0.1 to 1. For the increased value of correlation factor, the PSNR value decreases. For instance if the correlation factor is 0.3 and image size is 306x648 the ALGO A: Wavelets simulates the PSNR value equals to 47.96 and ALGO B: Contourlets gives 49.66 as PSNR value for the same correlation factor and image size.

Experiment No. 3: Image Name BIRD

		ALGO A: Wavelets			**ALGO B: Contourlets**		
Image Size	**Correlation Factor (α)**	**MSE (%)**	**Payload (%)**	**PSNR (dB)**	**MSE (%)**	**Payload (%)**	**PSNR (dB)**
256x256	0.1	0.331	32.7	50.69	0.281	32.7	52.19
	0.2	0.46	41.29	47.72	0.43	41.29	49.22
	0.3	0.62	45.43	44.13	0.59	45.43	45.63
	0.4	1.53	51.73	42.86	1.5	51.73	44.36
	0.5	1.85	55.69	38	1.82	55.69	39.5
	0.6	2.6	63.19	32.97	2.57	63.19	34.47
	0.7	3.71	65	29.68	3.68	65	31.18
	0.8	4.89	67.81	28.53	4.86	67.81	30.03
	0.9	6.16	69.62	24.36	6.13	69.62	25.86
	1	7.91	72.41	22.21	7.88	72.41	23.71
306x648	0.1	0.36	38.44	52.69	0.29	38.44	54.39
	0.2	0.59	42.75	50.92	0.52	42.75	52.62
	0.3	0.74	49.99	48.03	0.67	49.99	49.73
	0.4	1.3	56.91	46.34	1.23	56.91	48.04
	0.5	1.98	61.98	42.63	1.91	61.98	44.33
	0.6	2.83	65.9	39.6	2.76	65.9	41.3
	0.7	3.83	69.09	37.07	3.76	69.09	38.77
	0.8	4.99	71.67	34.87	4.92	71.67	36.57
	0.9	6.3	73.88	32.89	6.23	73.88	34.59
	1	7.96	74.86	29.98	7.89	74.86	31.68
512x512	0.1	0.41	42.39	55.66	0.32	42.39	57.56
	0.2	0.56	46.71	54.18	0.47	46.71	56.08
	0.3	0.79	51.1	51.64	0.7	51.1	53.54
	0.4	1.32	57.84	48.56	1.23	57.84	50.46
	0.5	2	63.4	44.39	1.91	63.4	46.29
	0.6	2.86	66.52	41.44	2.77	66.52	43.34
	0.7	3.79	71.64	38.67	3.7	71.64	40.57
	0.8	5.05	74.03	36.2	4.96	74.03	38.1
	0.9	6.37	76.92	34.74	6.28	76.92	36.64
	1	7.44	79.54	31.12	7.35	79.54	33.02

Table 4. 4: Image Name BIRD

In Table 4.4 image 'BIRD' is used for simulating results for comparison of ALGO A and ALGO B. Observe the image size 512x512 and correlation factor 0.8 for both the algorithms. PSNR value for ALGO A: Wavelets is 36.2 and ALGO B: Contourlets gives 38.1 as PSNR value. This experiment also proves the better performance of ALGO B over ALGO A.

Experiment No. 4: Image Name CAPS

		ALGO A: Wavelets			**ALGO B: Contourlets**		
Image Size	**Correlation Factor (α)**	**MSE (%)**	**Payload (%)**	**PSNR (dB)**	**MSE (%)**	**Payload (%)**	**PSNR (dB)**
256x256	0.1	0.34	30.13	49.86	0.29	30.13	51.36
	0.2	0.42	40	46.72	0.39	40	48.22
	0.3	0.68	44.98	43.63	0.65	44.98	45.13
	0.4	1.62	51.06	41.59	1.59	51.06	43.09
	0.5	1.72	55.37	37.74	1.69	55.37	39.24
	0.6	2.77	62.71	32	2.74	62.71	33.5
	0.7	3.51	64.87	29.33	3.48	64.87	30.83
	0.8	4.74	67.46	27.62	4.71	67.46	29.12
	0.9	6.19	69.34	23.67	6.16	69.34	25.17
	1	8	72.53	21.38	7.97	72.53	22.88
306x648	0.1	0.37	38.41	51.65	0.3	38.41	53.35
	0.2	0.55	42.71	49.78	0.48	42.71	51.48
	0.3	0.7	49.95	48.99	0.63	49.95	50.69
	0.4	1.26	56.87	46.3	1.19	56.87	48
	0.5	1.94	61.94	42.59	1.87	61.94	44.29
	0.6	2.79	65.86	39.56	2.72	65.86	41.26
	0.7	3.76	69.05	37.03	3.69	69.05	38.73
	0.8	4.95	71.63	34.83	4.88	71.63	36.53
	0.9	6.26	73.84	32.85	6.19	73.84	34.55
	1	7.92	74.82	28.94	7.85	74.82	30.64
512x512	0.1	0.42	42.4	55.67	0.33	42.4	57.57
	0.2	0.57	46.72	53.19	0.48	46.72	55.09
	0.3	0.8	51.11	52.17	0.71	51.11	54.07
	0.4	1.33	57.85	47.57	1.24	57.85	49.47
	0.5	2.01	63.58	45.2	1.92	63.58	47.1
	0.6	2.87	66.44	42.45	2.78	66.44	44.35
	0.7	3.9	69.05	38.67	3.81	69.05	40.57
	0.8	5.06	71.63	37.21	4.97	71.63	39.11
	0.9	6.68	73.84	34.76	6.59	73.84	36.66
	1	7.45	74.79	30.13	7.36	74.79	32.03

Table 4. 5: Image Name CAPS

It has been observed from Table 4.5 that image name 'CAPS' is used for simulating results and as the correlation factor increases the value of PSNR value deteriorates for all image sizes. For correlation factor

0.5 and image size 256x256 the ALGO A: Wavelets results PSNR value equals to 37.74. ALGO B: Contourlets results 39.24 as PSNR value for the same correlation factor and image size. The ALGO B: Contourlets gives better performance than ALGO A: Wavelets.

Experiment No. 5: Image Name MASK

		ALGO A: Wavelets			ALGO B: Contourlets		
Image Size	**Correlation Factor (α)**	**MSE (%)**	**Payload (%)**	**PSNR (dB)**	**MSE (%)**	**Payload (%)**	**PSNR (dB)**
256x256	0.1	0.35	30.28	50.67	0.3	30.28	52.17
	0.2	0.44	40.17	48.21	0.41	40.17	49.71
	0.3	0.57	44.83	45.02	0.54	44.83	46.52
	0.4	1.46	51.28	42.21	1.43	51.28	43.71
	0.5	1.92	55.69	39.67	1.89	55.69	41.17
	0.6	2.56	62.8	32.81	2.53	62.8	34.31
	0.7	3.47	64.72	30.97	3.44	64.72	32.47
	0.8	4.68	67.47	28.77	4.65	67.47	30.27
	0.9	6.19	69.21	24.32	6.16	69.21	25.82
	1	7.74	72.02	22.01	7.71	72.02	23.51
306x648	0.1	0.39	38.42	51.67	0.32	38.42	53.37
	0.2	0.57	42.73	50.2	0.5	42.73	51.9
	0.3	0.72	49.97	48.01	0.65	49.97	49.71
	0.4	1.28	56.89	46.32	1.21	56.89	48.02
	0.5	1.96	61.96	42.61	1.89	61.96	44.31
	0.6	2.81	65.88	39.58	2.74	65.88	41.28
	0.7	3.76	69.07	37.05	3.69	69.07	38.75
	0.8	4.97	71.64	34.85	4.9	71.64	36.55
	0.9	6.28	73.86	32.87	6.21	73.86	34.57
	1	7.94	74.54	28.96	7.87	74.54	30.66
512x512	0.1	0.421	42.41	54.68	0.331	42.41	56.58
	0.2	0.58	46.73	52.2	0.49	46.73	54.1
	0.3	0.81	51.12	49.18	0.72	51.12	51.08
	0.4	1.34	57.86	48.58	1.25	57.86	50.48
	0.5	2.02	63.46	44.21	1.93	63.46	46.11
	0.6	2.88	66.51	40.46	2.79	66.51	42.36
	0.7	3.91	71.64	38.68	3.82	71.64	40.58
	0.8	5.07	74.06	36.22	4.98	74.06	38.12
	0.9	6.39	76.94	34.73	6.3	76.94	36.63
	1	7.46	79.56	31.15	7.37	79.56	33.05

Table 4. 6: Image Name MASK

In Table 4.6 input cover image 'MASK' is used with increased values of correlation factor, which in turn decreases the PSNR value. If 512x512 image size is considered for comparisons with correlation factor 0.2, the ALGO A: Wavelets gives PSNR value equals to 52.2 and 54.10 as PSNR value using ALGO B: Contourlets for the same correlation factor and image size. It concludes that the contourlet transform results in better image quality than wavelet transform if the image size and payload remains constant.

Experiment No. 6: Image Name BOATS

		ALGO A: Wavelets			ALGO B: Contourlets		
Image Size	**Correlation Factor (α)**	**MSE (%)**	**Payload (%)**	**PSNR (dB)**	**MSE (%)**	**Payload (%)**	**PSNR (dB)**
256x256	0.1	0.38	32.12	49.73	0.33	32.12	51.23
	0.2	0.52	40	46.27	0.49	40	47.77
	0.3	0.67	45.13	43.08	0.64	45.13	44.58
	0.4	1.57	51.25	41.57	1.54	51.25	43.07
	0.5	1.9	55.78	37.78	1.87	55.78	39.28
	0.6	2.75	63.04	32.51	2.72	63.04	34.01
	0.7	3.52	64.49	29.14	3.49	64.49	30.64
	0.8	4.93	67.53	26.98	4.9	67.53	28.48
	0.9	6.19	69.27	23.39	6.16	69.27	24.89
	1	7.98	72.18	21.26	7.95	72.18	22.76
306x648	0.1	0.4	38.46	51.71	0.33	38.46	53.41
	0.2	0.61	42.77	49.84	0.54	42.77	51.54
	0.3	0.76	50.02	48.05	0.69	50.02	49.75
	0.4	1.32	56.93	46.42	1.25	56.93	48.12
	0.5	2.05	62.09	42.66	1.98	62.09	44.36
	0.6	2.84	65.92	39.62	2.77	65.92	41.32
	0.7	3.84	69.11	37.09	3.77	69.11	38.79
	0.8	4.83	71.69	34.89	4.76	71.69	36.59
	0.9	6.32	73.87	32.91	6.25	73.87	34.61
	1	7.98	74.88	27	7.91	74.88	28.7
512x512	0.1	0.44	38.51	55.76	0.35	38.51	57.66
	0.2	0.61	42.82	51.29	0.52	42.82	53.19
	0.3	0.83	51.07	50.1	0.74	51.07	52
	0.4	1.37	56.98	48.47	1.28	56.98	50.37
	0.5	2.1	62.14	43.71	2.01	62.14	45.61
	0.6	2.89	65.97	41.67	2.8	65.97	43.57
	0.7	3.93	69.16	38.14	3.84	69.16	40.04
	0.8	4.91	71.74	36.94	4.82	71.74	38.84
	0.9	6.37	73.92	32.96	6.28	73.92	34.86
	1	8.03	74.93	31.05	7.94	74.93	32.95

Table 4. 7: Image Name BOATS

Image 'BOATS' is used for simulating results in Table 4.7 for ALGO A: Wavelets and ALGO B: Contourlets. As correlation factor increases the PSNR value deteriorates for each image size. For correlation factor 0.7 and image size 256x256 the ALGO A: Wavelets gives PSNR value equals to 29.14. ALGO B: Contourlets gives 30.64 as PSNR value for the same correlation factor and image size. ALGO B has performed better than ALGO A in all the results presented in Table 4.7.

Experiment No. 7: Image Name BUS

		ALGO A: Wavelets			**ALGO B: Contourlets**		
Image Size	**Correlation Factor (α)**	**MSE (%)**	**Payload (%)**	**PSNR (dB)**	**MSE (%)**	**Payload (%)**	**PSNR (dB)**
256x256	0.1	0.26	32.94	51.61	0.21	32.94	53.11
	0.2	0.4	39.82	47.15	0.37	39.82	48.65
	0.3	0.55	45.01	45.16	0.52	45.01	46.66
	0.4	1.45	51.13	41.94	1.42	51.13	43.44
	0.5	1.78	55.47	38.63	1.75	55.47	40.13
	0.6	2.63	62.87	33.71	2.6	62.87	35.21
	0.7	3.6	64.69	30.24	3.57	64.69	31.74
	0.8	4.81	67.41	27.74	4.78	67.41	29.24
	0.9	6.07	69.12	25.26	6.04	69.12	26.76
	1	7.86	71.96	22.92	7.83	71.96	24.42
306x648	0.1	0.29	38.32	52.57	0.22	38.32	54.27
	0.2	0.47	42.63	50.46	0.4	42.63	52.16
	0.3	0.62	49.87	48.91	0.55	49.87	50.61
	0.4	1.186	56.79	46.22	1.116	56.79	47.92
	0.5	1.86	61.84	42.51	1.79	61.84	44.21
	0.6	2.71	65.78	39.43	2.64	65.78	41.13
	0.7	3.74	78.94	36.95	3.67	78.94	38.65
	0.8	4.81	71.55	34.75	4.74	71.55	36.45
	0.9	6.82	72.64	32.37	6.75	72.64	34.07
	1	7.44	74.28	27.86	7.37	74.28	29.56
512x512	0.1	0.33	38.63	55.3	0.24	38.63	57.2
	0.2	0.51	42.69	52.16	0.42	42.69	54.06
	0.3	0.68	49.93	51.97	0.59	49.93	53.87
	0.4	1.24	56.85	46.28	1.15	56.85	48.18
	0.5	1.92	61.92	42.57	1.83	61.92	44.47
	0.6	2.74	65.84	39.54	2.65	65.84	41.44
	0.7	3.74	69.03	37.01	3.65	69.03	38.91
	0.8	4.86	71.61	35.21	4.77	71.61	37.11
	0.9	6.29	73.82	32.83	6.2	73.82	34.73
	1	7.9	74.73	30.05	7.81	74.73	31.95

Table 4. 8: Image Name BUS

In Table 4.8, it has been observed that for image name 'BUS' the simulated results have been calculated to give a inverse relationship of correlation factor on PSNR value. For image size 512x512 and correlation factor 0.7 the ALGO A: Wavelets gives 37.01 as PSNR value. Same input parameters have been used for ALGO B: Contourlets to compare the performance and it results PSNR value equals to 38.9. From the above given comparison, the performance of ALGO B is far better than ALGO A, as both algorithms used different transforms.

Experiment No. 8: Image Name TREES

		ALGO A: Wavelets			**ALGO B: Contourlets**		
Image Size	**Correlation Factor (α)**	**MSE (%)**	**Payload (%)**	**PSNR (dB)**	**MSE (%)**	**Payload (%)**	**PSNR (dB)**
256x256	0.1	0.28	31.02	49.63	0.23	31.02	51.13
	0.2	0.42	39.9	47.17	0.39	39.9	48.67
	0.3	0.57	45.03	44.98	0.54	45.03	46.48
	0.4	1.47	51.15	41.17	1.44	51.15	42.67
	0.5	1.8	55.46	37.62	1.77	55.46	39.12
	0.6	2.65	62.89	32.77	2.62	62.89	34.27
	0.7	3.62	64.68	29.93	3.59	64.68	31.43
	0.8	4.83	67.43	27.73	4.8	67.43	29.23
	0.9	6.09	69.17	24.28	6.06	69.17	25.78
	1	7.89	71.98	21.97	7.86	71.98	23.47
306x648	0.1	0.3	38.33	51.58	0.23	38.33	53.28
	0.2	0.48	42.64	49.91	0.41	42.64	51.61
	0.3	0.63	49.88	47.92	0.56	49.88	49.62
	0.4	1.19	56.8	45.23	1.12	56.8	46.93
	0.5	1.87	61.87	42.52	1.8	61.87	44.22
	0.6	2.69	65.79	39.49	2.62	65.79	41.19
	0.7	3.69	68.98	36.96	3.62	68.98	38.66
	0.8	4.81	71.56	35.16	4.74	71.56	36.86
	0.9	6.24	73.77	32.78	6.17	73.77	34.48
	1	7.85	74.68	26.87	7.78	74.68	28.57
512x512	0.1	0.36	33.05	54.72	0.27	33.05	56.62
	0.2	0.51	39.94	51.26	0.42	39.94	53.16
	0.3	0.66	47.92	50.07	0.57	47.92	51.97
	0.4	1.53	52.24	48.27	1.44	52.24	50.17
	0.5	1.89	55.58	44.74	1.8	55.58	46.64
	0.6	2.74	63.98	41.82	2.65	63.98	43.72
	0.7	3.71	65.8	39.06	3.62	65.8	40.96
	0.8	4.92	67.52	37.84	4.83	67.52	39.74
	0.9	6.18	69.23	34.37	6.09	69.23	36.27
	1	7.97	72.18	32.03	7.88	72.18	33.93

Table 4. 9: Image Name TREES

Now consider the image size 306x648 of image 'TREES' and correlation factor value 1 in table 4.9. The PSNR value for ALGO B: Contourlets is improved than PSNR value for ALGO A: Wavelets. In Table 4.9, the correlation factor has been increased from 0.1 to 1 for observing the affect on PSNR value while keeping the payload same for both the algorithms to evaluate the performance. For the given correlation factor and image size, the ALGO A: Wavelets gives PSNR value equals to 26.87. ALGO B: Contourlets gives 28.57 as PSNR value for the same correlation factor and image size. ALGO B consistently performs well than ALGO A for each image size.

Experiment No. 9: Image Name BUILDING

		ALGO A: Wavelets			**ALGO B: Contourlets**		
Image Size	**Correlation Factor (α)**	**MSE (%)**	**Payload (%)**	**PSNR (dB)**	**MSE (%)**	**Payload (%)**	**PSNR (dB)**
256x256	0.1	0.4	34.14	50.75	0.35	34.14	52.25
	0.2	0.54	40.02	48.29	0.51	40.02	49.79
	0.3	0.69	45.15	45.1	0.66	45.15	46.6
	0.4	1.9	51.27	42.59	1.87	51.27	44.09
	0.5	1.92	55.82	38.8	1.89	55.82	40.3
	0.6	2.77	63.06	32.53	2.74	63.06	34.03
	0.7	3.54	64.51	30.16	3.51	64.51	31.66
	0.8	4.95	67.55	27.4	4.92	67.55	28.9
	0.9	6.21	69.29	24.41	6.18	69.29	25.91
	1	8	72.21	22.28	7.97	72.21	23.78
306x648	0.1	0.41	38.49	51.74	0.34	38.49	53.44
	0.2	0.64	42.8	49.27	0.57	42.8	50.97
	0.3	0.79	50.04	47.09	0.72	50.04	48.79
	0.4	1.35	56.96	45.39	1.28	56.96	47.09
	0.5	2.01	62.03	42.68	1.94	62.03	44.38
	0.6	2.84	65.95	39.65	2.77	65.95	41.35
	0.7	3.8	69.14	37.12	3.73	69.14	38.82
	0.8	4.71	71.72	34.92	4.64	71.72	36.62
	0.9	6.29	73.93	32.94	6.22	73.93	34.64
	1	7.92	74.91	28.03	7.85	74.91	29.73
512x512	0.1	0.46	32.21	54.82	0.37	32.21	56.72
	0.2	0.63	40.09	52.36	0.54	40.09	54.26
	0.3	0.76	49.56	51.17	0.67	49.56	53.07
	0.4	1.66	53.34	48.73	1.57	53.34	50.63
	0.5	1.93	55.57	43.87	1.84	55.57	45.77
	0.6	2.84	64.13	42.96	2.75	64.13	44.86
	0.7	3.81	65.58	39.12	3.72	65.58	41.02
	0.8	5.02	67.62	37.47	4.93	67.62	39.37
	0.9	6.28	69.36	34.48	6.19	69.36	36.38
	1	8.07	72.17	32.09	7.98	72.17	33.99

Table 4. 10: Image Name BUILDING

Repeating the same performance evaluation experiments for the image 'BUILDING' of size 256x256 and correlation factor 0.4, the ALGO A: Wavelets gives PSNR value equals to 42.59 and ALGO B: Contourlets gives 44.09 as PSNR value for the same correlation factor and image size. PSNR value deteriorates as correlation factor increases and keeping the image sizes same. The comparisons for ALGO A and ALGO B for image name 'BUILDING' have been tabulated in Table 4.10. It has been concluded that ALGO B gives constantly better performance than ALGO A.

Experiment No. 10: Image Name STAIRS

		ALGO A: Wavelets			**ALGO B: Contourlets**		
Image Size	**Correlation Factor (α)**	**MSE (%)**	**Payload (%)**	**PSNR (dB)**	**MSE (%)**	**Payload (%)**	**PSNR (dB)**
256x256	0.1	0.27	32.01	49.62	0.22	32.01	51.12
	0.2	0.43	39.89	46.16	0.4	39.89	47.66
	0.3	0.56	45.02	43.97	0.53	45.02	45.47
	0.4	1.46	51.14	40.46	1.43	51.14	41.96
	0.5	1.79	55.37	37.67	1.76	55.37	39.17
	0.6	2.64	62.93	32.76	2.61	62.93	34.26
	0.7	3.61	64.38	29.92	3.58	64.38	31.42
	0.8	4.82	67.42	27.27	4.79	67.42	28.77
	0.9	6.08	69.16	24.28	6.05	69.16	25.78
	1	7.87	71.97	21.91	7.84	71.97	23.41
306x648	0.1	0.28	38.3	52.58	0.21	38.3	54.28
	0.2	0.45	42.61	50.08	0.38	42.61	51.78
	0.3	0.6	49.87	47.89	0.53	49.87	49.59
	0.4	1.16	56.72	46.2	1.09	56.72	47.9
	0.5	1.84	61.82	42.49	1.77	61.82	44.19
	0.6	2.69	65.76	39.46	2.62	65.76	41.16
	0.7	3.72	78.93	36.93	3.65	78.93	38.63
	0.8	4.79	71.52	34.73	4.72	71.52	36.43
	0.9	6.8	72.62	32.35	6.73	72.62	34.05
	1	7.42	74.26	27.84	7.35	74.26	29.54
512x512	0.1	0.33	38.41	55.62	0.24	38.41	57.52
	0.2	0.52	42.68	52.15	0.43	42.68	54.05
	0.3	0.67	49.92	51.96	0.58	49.92	53.86
	0.4	1.24	56.84	48.27	1.15	56.84	50.17
	0.5	1.91	61.91	45.56	1.82	61.91	47.46
	0.6	2.76	66.83	41.53	2.67	66.83	43.43
	0.7	3.71	69.02	40.03	3.62	69.02	41.93
	0.8	5.1	71.59	36.8	5.01	71.59	38.7
	0.9	6.23	73.81	33.82	6.14	73.81	35.72
	1	7.68	74.49	32.13	7.59	74.49	34.03

Table 4. 11: Image Name STAIRS

It has been observed from Table 4.11 that image name 'STAIRS' is used for simulating results and as we keep on increasing the correlation factor the value of PSNR value deteriorates for each image size. For correlation factor 0.2 and image size 306x648 the ALGO A: Wavelets gives PSNR value equals to 50.08. ALGO B: Contourlets gives 51.78 as PSNR value for the same correlation factor and image size.

It is clear from the above simulated results that standard trend has been followed. As we increase the correlation factor for any image of a size, the MSE value increases and which in turn decreases the value of PSNR.

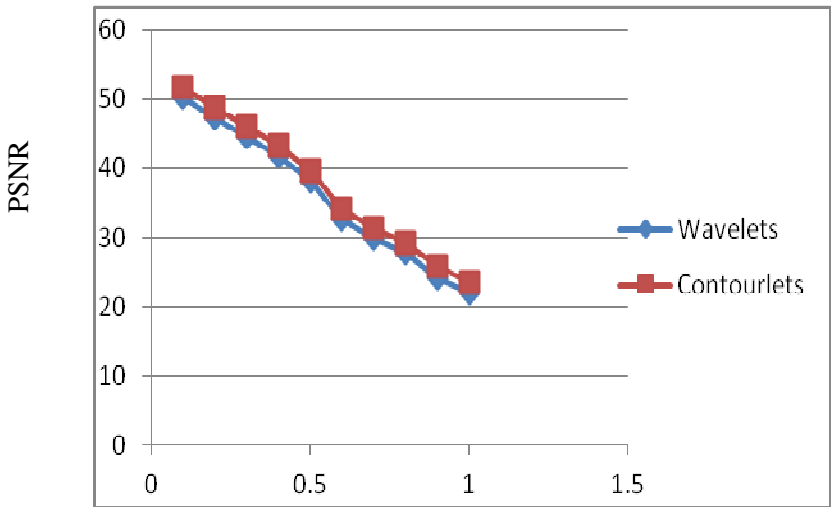


Figure 4.1: 256x256

In Figure 4.1, the results of ALGO A and ALGO B have been compared for same image size. The contourlet transform consistently performs well than wavelet transform. For the justification, the image name 'PATTERN' of size 256x256 and correlation factor is 0.4, the PSNR value computed using ALGO A: Wavelets is 41.21 and if we compute it using ALGO B: Contourlets, the PSNR is 42.71. In this comparison, the correlation has an inverse relation with PSNR value.

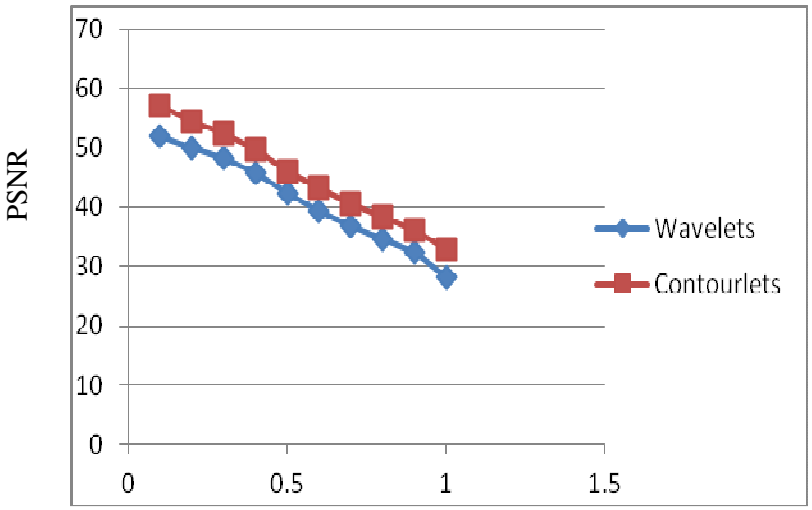


Figure 4.2: 306x648

Now we have compared the performance for image size 306x648 using the ALGO A and ALGO B. It also concludes that contourlet performs well than wavelet transform if we execute both algorithms on same image of same size. Figure 4.2 illustrates the performance evaluation for both transforms. In Figure 4.3, the 512x512 image size has been taken. If image size is same for each algorithm, the performance of ALGO B for contourlet transform performs well, then by increasing image size for both algorithms will not affect the performance of contourlets.

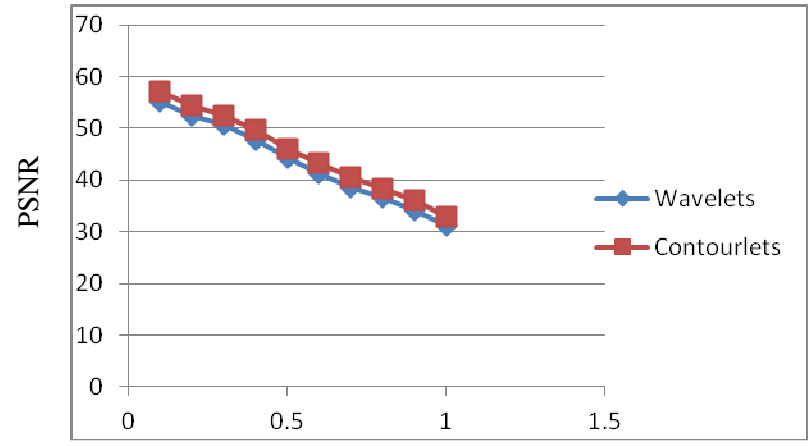


Figure 4.3: 512x512

The results of Ali Al-Taby's work have been compared with the proposed ALGO A using wavelets and ALGO B using contourlets. Figure 4.4 illustrates the clear increase of PSNR values with respect to the correlation factor for ALGO A comparison.

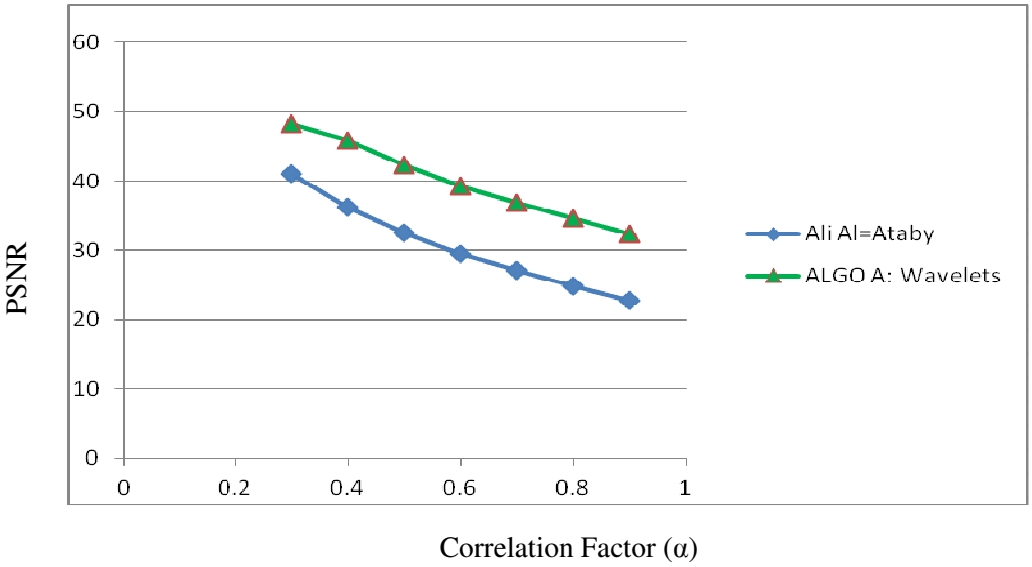


Figure 4.4: Comparisons of PSNR values for ALGO A

Figure 4.5 gives a clear picture of the old results reported by Ali Al-Taby[62] with ALGO B. In this the contourlet transform results have been compared. As we already concluded that contourlets gives us better PSNR values in comparison with wavelets. It is very obvious that contourlets performs better than Ali Al-Taby's work where wavelets were used.

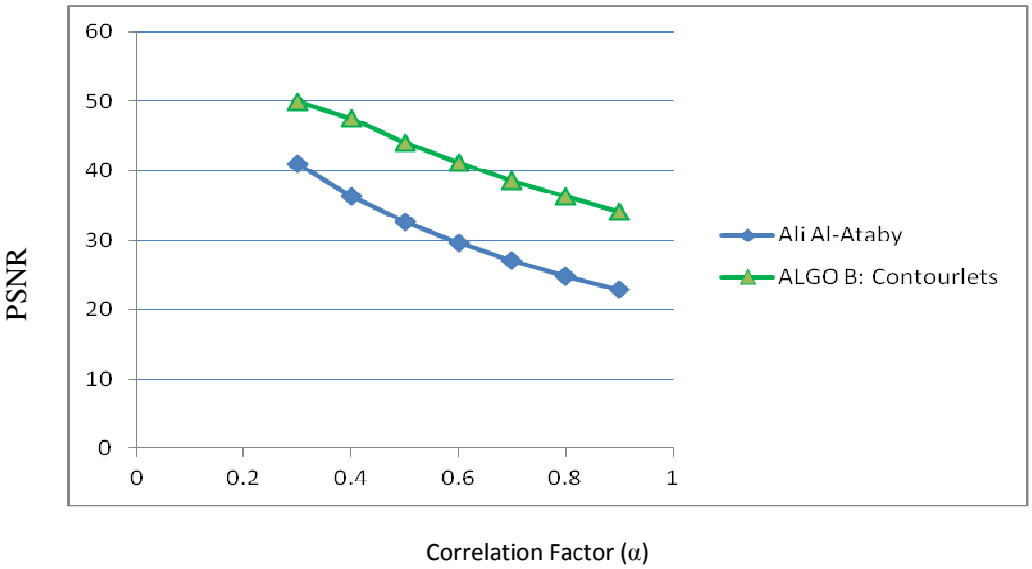


Figure 4.5: Comparisons of PSNR values of ALGO B

In Figure 4.6, the ALGO A for wavelets has been compared with ContSteg [56] for different number of characters. Here the payload is increased keeping the correlation factor constant and it directly affects the PSNR value.

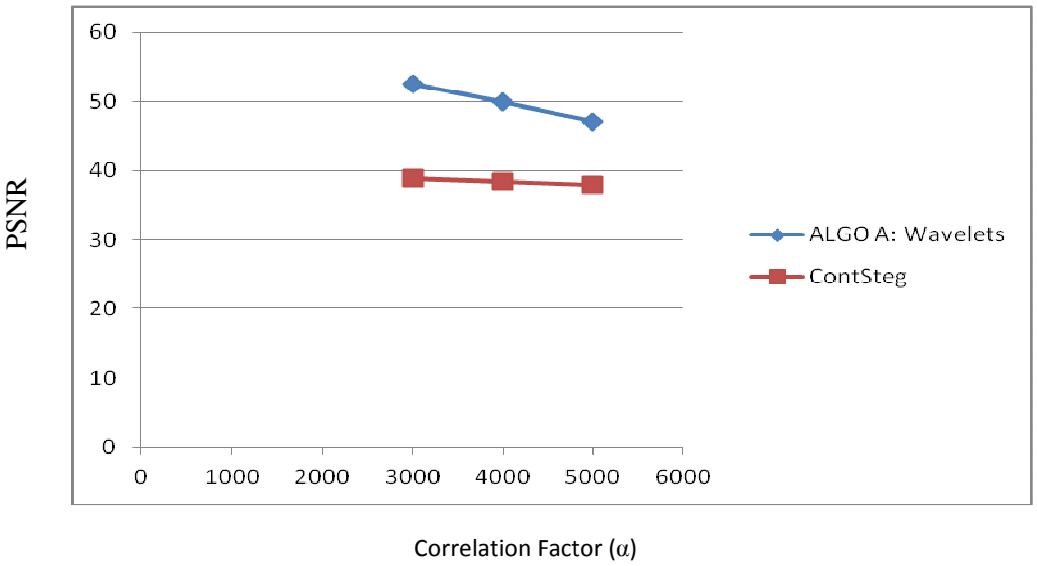


Figure 4.6: Comparison of ALGO A: Wavelets with [56]

In Figure 4.7, the ALGO B for contourlets has been compared with ContSteg [56] for different number of characters. In this comparison, as the number of characters increases the PSNR value deteriorates.

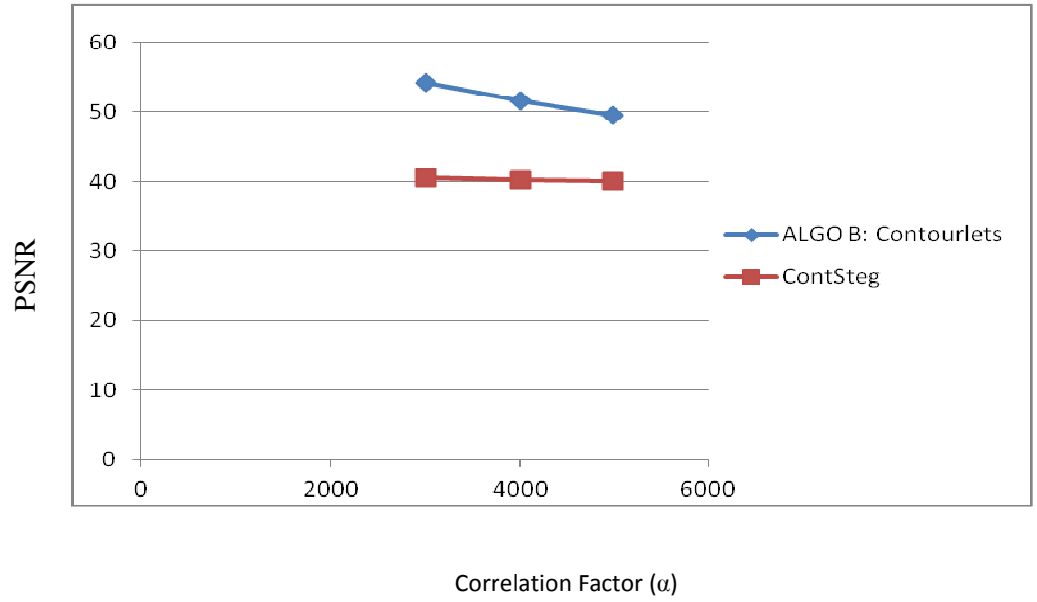


Figure 4.7: Comparison of ALGO B: Contourlets with [56]

4.4 Results of ALGO C

Results have been obtained using a Microsoft Word file of varying number of characters. This work has been simulated using jpeg format colored cover images of size 512x512. The simulation was carried out using three images of same size. First PSNR was calculated for a single image using three different sizes of message files. In Table 4.12, the results of simulation have been shown. As we take a standard image BOATS and number of character are 4915, the PSNR value is 63.46 in dB. But if we increase the number of characters to 6809, the PSNR value decreases because we are trying to encode more information. Table 4.12 shows the results of ALGO C:-

Image Name	No of Characters	PSNR (dB)
LENA	3584	64.79
	4915	63.46
	6809	60.53
BOATS	3584	64.27
	4915	63.07
	6809	60.36
MASK	3584	63.84
	4915	62.96
	6809	60.1

Table 4. 12: ALGO C Simulation Results

The above simulated results are now presented in graphical form in Figure 4.8. Here we can clearly observe the trend of increasing the number of characters will decrease image quality and it reduces the image imperceptibility or deteriorates the PSNR values.

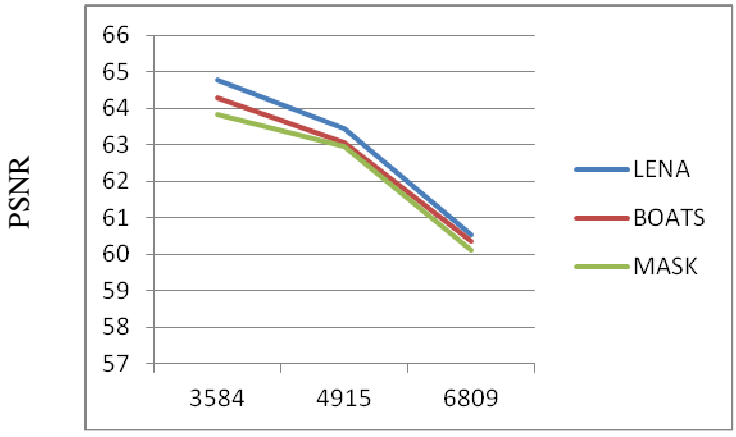


Figure 4.8: ALGO C

In Figure 4.9 a comparison of ALGO C with Amitav Nag's proposed method in [59] has been shown. A clear picture of performance evaluation has been presented. The PSNR for LENA and BOATS have been calculated using 5400 number of characters. Both the methods decrease the PSNR values as more information is encoded. But ALGO C performs 4-5% better than Amitav et. al. method [59].

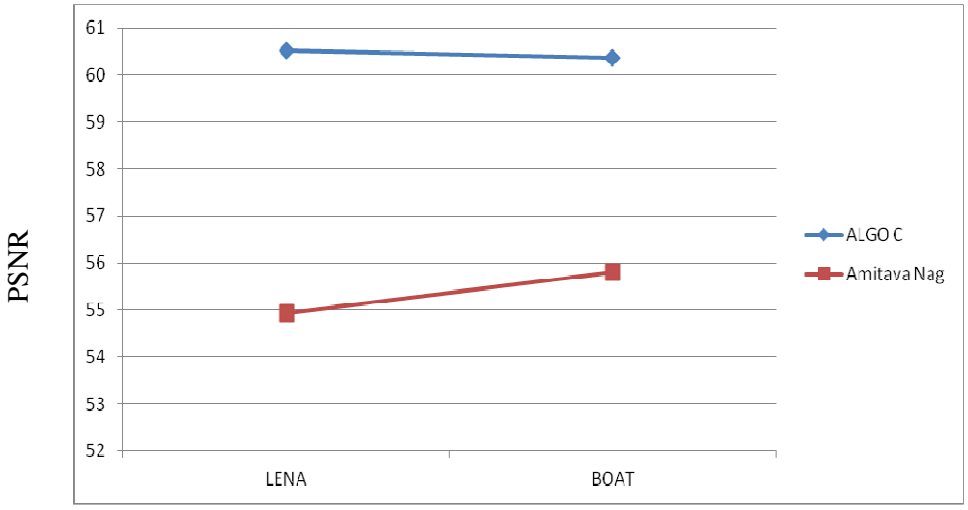


Figure 4.9: Comparison of ALGO C with Amitav Nag[59]

It is pertinent to note that the ALGO C has outperformed Amitav Nag's method with a wide margin.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

Steganography is a relatively new field. The major objective of using steganography is to hide information in some other medium so as to provide protection against detection. Here, the success of a technique lies in changing the cover medium in a way that makes the change imperceptible for both the human and computer. It differs from cryptography because cryptography aims at concealing the secret message. A major limitation of steganography is the size of the cover medium used. Therefore, all the Steganographic methods aim to achieve higher hiding capacity in a given medium.

Major media that can be used for steganography are text, image, audio and video files. Most of the techniques still rely on the agreement between the sender and the receiver on the exchange of the key. There are quite a few ways in which text files could be used for information hiding (e.g. moving the text lines up or down, altering or changing the space between the words in the text file etc). Using the audio files, sample quantization, temporal sampling or perceptual sampling could be used to achieve information hiding.

Images however, offer a wide variety of ways in which additional information could be embedded. The major dimensions in which images can be used are spectrum, LSB substitution and transformed domain steganography. In LSB, the least significant bit of each pixel is replaced with the message bit. In transformed domain steganography, data hiding is achieved by Fourier transform, Discrete cosine transform, Wavelet Transform, Contourlet Transform. These techniques are more robust and difficult to detect as compared to LSB insertion.

There are few techniques available for color image steganography. This is because, most of the color image formats like jpeg are larger in size and communicating heavy images over open systems like internet can raise suspicions. One of the techniques suggests using edge pixels to hide the message bits.

The advantage of using this technique over LSB is that an edge pixel even if altered, will not stand out extraordinarily among its neighboring pixels. Another method uses variable bits from the RGB channels for hiding information. Block difference methods have also been proposed. However, the selection of blocks could be done in a variety of ways.

Encryption algorithms have also been discussed and these can be categorized into secret key encryption, public key encryption and hash functions, based on the number of keys used for encryption and decryption. Blowfish encryption algorithm has been selected for the reported better performance in comparison with other secret key encryption algorithms.

The technique adopted in the research work uses the wavelet transform and contourlet transform. Three steganographic systems have been proposed for improved image steganography using different image sizes of JPEG format. The first proposed system using wavelet first, preprocessed the image and then performs the wavelet decomposition to level 3. It encodes the encrypted data from blowfish algorithm. The second proposed system functions similar to the first one but it uses the contourlet transform. The performance evaluations have been carried out for comparing both the transforms for image steganographic systems. The third proposed system which uses Huffman encoding scheme also gives improved results in terms of number of characters encoded and PSNR values.

The proposed systems have been compared with other researcher's work and proved that the proposed system are improved in terms of PSNR values and image imperceptibility. Moreover, there is a very limited work done in wavelet and contourlet transform domain for colored image steganography.

It has also been discussed that the proposed systems inherently preserves the image quality and operates well above on PSNR scale in all the experiments performed. Currently, only one page information of .doc/.docx file is being used. However, the algorithms can be modified to use more number of pages or more number of characters.

5.2 Limitations

Currently, the proposed system takes a lot of time and performs very slow. High performance computing can be used to implement the same proposed algorithms to test its time efficiency.

5.3 Future Work

Currently the algorithm which makes use of Huffman encoding can be modified to encode a large amount of data by exploiting all the available bits in each RGB component. It can increase the capacity of the system. Larger sizes of texts can be hidden and using more bits without compromising quality of cover image.

Time is another important requirement for a generic steganographic system but it can easily be compromised to fulfill the other three requirements i.e. capacity, robustness and imperceptibility. To achieve time efficiency for steganographic systems, cilk and other high performance computing languages can be involved without compromising the main requirements.

Texture images can be used for contourlets to have better image quality. As texture images have many edges and contours, appropriate methodologies and experiments can provide better results while using texture images.

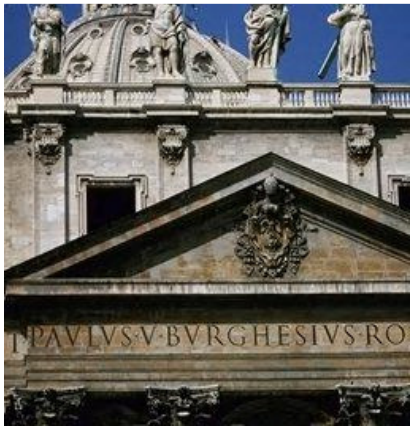
Appendix ‘A’

Input Cover Images

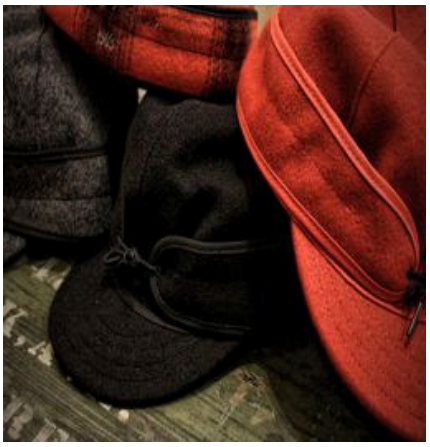
BIGCATS

BIRD

BOATS

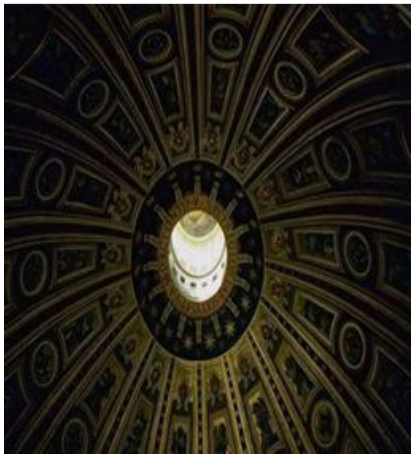


BUILDING



CAPS

MASK



PATTERN

BUS

TREES

STAIRS

Appendix 'B'

Accepted Journal Papers

1. Accepted in International Journal of Computer Science Issues, Volume 9, Issue 2, March 2012

Improved Image Steganography Technique for Colored Images using Huffman Encoding with Symlet Wavelets

Saddaf Rubab[1], Dr. M. Younus[2]

[1&2]Department of Computer Engineering, College of Electrical & Mechanical Engineering,
National University of Sciences & Technology (NUST), Islamabad, Pakistan
sdf_rubab@hotmail.com[1],
myjaved@ceme.nust.edu.pk

Abstract
Steganography is data hiding technique that is mostly used. It uses cover object of type text, images, and videos. The paper presents a new devised algorithm to hide text in any colored image of any size using Huffman encryption and 2D Wavelet Transform. We presented simulated results which prove that there is very negligible image quality degradation. We used PSNR metric for this purpose. The subject algorithm also proved secure as Huffman table is required to decode the information.

Key Words: *Image steganography, Huffman, Wavelet transform*

1. Introduction

We have many techniques to imply data hiding including cryptography, steganography and watermarking. In this paper steganography is discussed only. Steganography is evolved from two Greek words that mean "covered writing". It is used to hide data within anther one either of same type or of different. It maintains the beauty of data hiding by providing better security and imperceptibility of hidden message in cover object. The main process of steganography is done by hiding the secret content in cover media by using any steganography method, so that any other party or enemy is not able to even guess that any message is embedded in [1]. To get back the secret message the reverse of steganography, called steganalysis is used. Figure 1 explains the above in pictorial form.

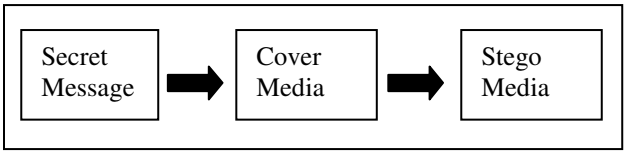


Figure 1: Image Steganography Block Diagram

Encryption is different from steganography. Encryption involves plain text and convert it to encrypted or cipher text using unique encryption key [2]. Encrypted text can only be converted back to plain text if key is known to the recipient. In both encryption and steganography we hide our original message, but output of encryption is detectable by reader and output of steganography is not detectable by reader by just seeing the text. It is clear from the difference that encrypted text is more vulnerable to security attacks by opponents than steganography.
Steganography use different cover media like, text, image, video etc. Image is most common cover media for it and we call it Image Steganography. We divide it in two domains:

a. Spatial Domain
b. Transform Domain

In spatial domain secret messages are inserted in the Least Significant Bit (LSB) of image or by bit shifting. In transform domain, secret message is embedded in the coefficients of cover media in frequency domain. In transform domain cosine, wavelet etc are used as transforms. Section 2 explains wavelet transform in more detail and Huffman algorithm in Section 3. Devised method is explained in Section 4 and simulated results are shown in Section 5. Conclusions and future work are illustrated in Section 6.

2. Wavelet Transform

In comparison to other transforms, wavelet transforms proved to be the best for image transformation [3]. In its basic operations, it decomposes the input signal into set of functions which are called wavelets. For image applications in transform domain, wavelet transform of image is computed, then modifications are made and at final step, inverse of wavelet is taken to get resulted image. We can select any family of wavelet from discrete or continuous wavelets like Haar, Coiflet,Symlet, Daubechies [4]. In discrete wavelets, we have different levels like 1-D, 2-D … n-D. This work presents 2D wavelet transform and symlet wavelets. Original signal is decomposed twice in 2-DWT in a way that makes use of scaling and wavelet functions of level 1 or 1-DWT. Figure 2 explains it in detail.

3. Huffman (Encryption Algorithm)

Several encryption algorithms including DES [5], AES [6], Triple DES [5], were used in past few years. But variable length codes get more attention from researchers in little time. It is a code that maps source symbols to variable length of bits like Lempel-Ziv coding, Huffman codes and arithmetic coding.

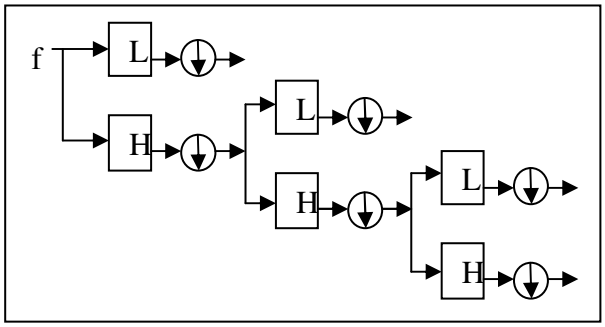


Figure 2: 2DWT

1952 [7] when he was a PhD student at MIT. It is used for lossless data compression. It represents the data in few bits and in few memory locations. It creates binary tree. The process starts by set of symbols/letters and their respective frequencies in ascending or descending order. Each symbol with its frequency is a leaf node at start. Next step is to select two symbols with smallest frequencies, add their frequencies and assign it to parent node, until only one node remains which is called the root node. Assign 0's and 1's to all the nodes and translate the codes by reading from leaf to root node. This way we construct a Huffman Table.

3.1 Huffman Table

The second part of Huffman is to decode the tree generated in compression of data. In this, we create Huffman table, which is used to decode symbols/letters in original data using the generated codes. That's the reason it implements more security because Huffman table is required at the recipient side. It prevents enemy attacks on secret data.

4. Proposed Steganography Method

The proposed algorithm was implemented in MATLAB 7.9. It is divided into two main phases. Following are the steps of devised method:

START
Phase I: Huffman Encoding
Step I: Obtain secret Message
Step II: Perform Huffman encoding on the secret message.
Step III: Create Huffman Table.
Step IV: Perform the binary conversion.
Step V: Make groups of 3bits each
Check last group (if number of bits! = 3)
Add zero at end of bits (0)
End
End
Phase II:
Step I: Select colored Cover Image.
Step II: Separate Red, Green and Blue Components (RGB).
Step III: Apply 2DWT on each component separately.
Step IV: Embed Huffman codes (calculated in Phase I)
Step V: Add RGB Components
Step VI: Apply Inverse 2DWT
End
END

To embed Huffman codes, firstly divide Huffman groups by 3. For example we have 12 groups formed by Huffman encryption phase. By dividing it by three (03), we get 3 sets of 4 groups each. Now select 3 consecutive bits of first set and insert into the Least Significant Bit (LSB) of 2DWT coefficient of red component. Do this for the rest of three groups in set 1. Repeat this with set 2 for 2DWT coefficient of green component of and set 3 for 2DWT coefficients of blue component.

5. Experimental Results

5.1 Peak Signal To Noise Ratio (PSNR)

In this work PSNR values are used to show the image imperceptibility [8].

$$MSE = \frac{1}{MN} \sum_{i=0}^{M} \sum_{j=0}^{N} (Xij - \overline{Xij}) \quad (1)$$

$$PSNR\ (dB) = 10\log_{10} \left(\frac{I^2}{MSE}\right) \quad (2)$$

X_{ij} represents the i^{th} rows and j^{th} columns of original image and $\overline{X_{ij}}$ represents the i^{th} rows and j^{th} columns of transformed image. Higher the PSNR value means more difficult to perceive that any hidden message is hidden.

5.2 Results

Devised method is implemented in MATLAB 7.9 using Windows 7 platform. Results are carried out with a Microsoft Word file of 1 page varying number of letters. This work is simulated using jpeg format colored cover images of size 512x512. Following are the three images of size 512x512 to discuss the results.

Figure 3a: Lena

Figure 3b: Boats

Figure 3c: Mask

Table 1 shows the PSNR values against the numbers of characters embedded. It is clear from the results that by increasing the number of characters the PSNR value decays. All the images are of same size, if we increase image size and number of characters remains constant, then the PSNR value will improve.

6. Conclusions

The presented algorithm is for any size of colored image. It gives more capacity for larger image sizes. It enhances security and also preserves the image quality. By inserting Huffman codes into the three components of colored image it becomes complicated but it provides us with better security measures, we can say it provides triple security. In terms that someone if find Huffman table it is not possible even then to decode the full message. This new method also improves the PSNR values. In future work may be carried out to increase the capacity and encode the secret message with more number of words.

Table 1: PSNR values for 512x512 images with respect to number of characters

Cover Image (.jpeg)	Number of characters	PSNR
Lena	3584	64.79
	4915	63.46
	6809	60.53
Boats	3584	64.27
	4915	63.07
	6809	60.36
Mask	3584	63.84
	4915	62.96
	6809	60.10

7. References

[1] N. Provos, P. Honeyman, "Hide and seek: An introduction to steganography", IEEE Security Privacy Magazine (2003), Volume: 1, Issue: 3, Publisher: IEEE Security & Privacy, Pages: 32-44

[2] Postnote October 2006 Number 270 Data encryption

[3]http://scholar.lib.vt.edu/theses/available/etd-12062002-152858/unrestricted/Chapter4.pdf

[4] C. E. Heily and D. F. Walnut, "Continuous and discrete wavelet transforms", SIAM Review 1989 Society for Industrial and Applied Mathematics, Vol. 31, No. 4, pp. 628-666, December 1989 007

[5] National Institute of Standards and Technology, "Data Encryption Standard (DES)", FIPS PUB 46-3, 1999 October 25

[6] NIST, "Report on the Development of the Advanced Encryption Standard (AES)", October 2, 2000

[7] David A. Huffman, “A Method for the Construction of Minimum - Redundancy Codes”, Proceedings of the I.R.E., September 1952, pp 1098–1102
[8] Almohammad, A.; Ghinea, G.; “Stego image quality and the reliability of PSNR”, 2nd International Conference on Image Processing Theory Tools and Applications (IPTA), 2010, Pages: 215 – 22

2. Accepted in International Journal of Computer Applications, February 2012 Edition

Improved Image Steganography Technique for Colored Images using Wavelet Transform

Saddaf Rubab[1]
[1]Department of Computer Engineering, College of Electrical & Mechanical Engineering, National University of Sciences & Technology (NUST), Islamabad, Pakistan
sdf_rubab@hotmail.com

Dr. M. Younus[2]
[2]Department of Computer Engineering, College of Electrical & Mechanical Engineering, National University of Sciences & Technology (NUST), Islamabad, Pakistan
myjaved@ceme.nust.edu.pk

ABSTRACT

Steganography is the most used technique for data hiding. We can implement it using any cover media like text, images, and videos. In this paper we devised a new algorithm to hide our text in any colored image of any size using wavelet transform. It improves the image quality and imperceptibility. Our method sustains the security attacks. Extensive testing is performed using different sizes of images and presented our results in payload and PSNR values.

General Terms

Steganography, payload, wavelet, transforms, stego image, colored image, blowfish

INTRODUCTION

Data hiding can be done by cryptography, steganography and watermarking. We are here only considering steganography, which literally means "covered writing". The steganography is science of data hiding within another one, so that its presence is undetectable and suffers less security threats or attacks. It hides the content in cover media as not to provoke any doubt that there is some information or message hidden in the media [1]. Sometimes people mix it with encryption. In encryption the content is not hidden but not readable by the reader if the key is not known to him. But the encrypted content can be intercepted by anyone and chances always present that he will try to decode it or affect it by attempting to decode it for a purpose or just for the sake of curiosity. Whereas, steganography gives us more freedom to communicate and send secret information without leaving any evidence that opponent will intercept and try decoding your information.

For this, different cover media can be used like, text, image, video etc. The content to be covertly sent is payload which is called stego text after application of any steganographic technique and media used is called stego image/text/video/protocol (depending on the choice of media you are using). The opposite of it is steganalysis, which is out of the scope of this paper.

The most popular cover object is image to perform steganography. Image steganography is divided into spatial and transform domain. In spatial domain messages are embedded in the intensity of image pixel like in LSB. Whereas in transform domain, image is first transformed and them message is encoded like discrete cosine transform (DCT), discrete wavelet transform (DWT) and many others.
Many different image file formats exists, but jpeg format proved to be the best among all [2].

WAVELET TRANSFORM

Wavelet transform gives the best result for image transformation [6]. It decomposes signal into a set of basic functions. There are two flavors of wavelet transform, one is discrete and other is continuous. We here focus on discrete wavelet transform. In DWT we have 1-D, 2-D… n-D levels. 2DWT is used in our work. It uses the scaling and wavelet functions of 1DWT. Figure 1 illustrate the 2DWT level. We can increase levels at the cost of complexity. It is two times decomposition of original signal via sub-division. to use any wavelet we have to select the filters and families like Coiflets, Haar, Symlet. We have selected Symlet 6 for our experimental work.

BLOWFISH (ENCRYPTION ALGORITHM)

It was designed by a cryptologist named Bruce Schneier and made it accessible for public. It is a 64-bit block cipher and variable key length. It has two parts. A first deal with the key expansion and second part performs the encryption of information. It increases contrast value in image by reducing the redundant information. In [5], it is presented that blowfish performs outclass compared to other encryption algorithms like AES, DES. We have selected blowfish to fulfill need of encrypted information.

RELATED WORK

Amitava Nag, Sushanta Biswas, Debasree Sarkar & Partha Pratim Sarkar in [7] presented a novel technique using DWT transform for the cover image transformation and then Huffman is used on secret message before embedding. It uses only high frequency coefficients for embedding message bits and neglected low ones to get better image quality.

Another image steganographic method using wavelet and Microsoft Utility for RC4 encryption [8] proposed which proves to be more secure.

M. F. Tolba, M. A. Ghonemy, I. A. Taha, and A. S. Khalifa in 2004 [9] utilizes wavelet transforms that map integers to integers and proposed an algorithm that embeds the message bitstream into the LSB's of the integer wavelet coefficients of a true-color image. Experimental results showed the high invisibility of the proposed model even with large message size.

PROPOSED STEGANOGRAPHY METHOD

Our work is improvement of results presented in [4]. My proposed algorithm was implemented in MATLAB 7.9 and Visual Studio 2010. My devised method consists of two main processes. First one deals with the MATLAB implementation which passes some controls to Visual Studio 2010 for C# implementation of blowfish algorithm. The other process then returns back the control to MATLAB from Visual Studio 2010. I have developed main steganographic routines in MATLAB using the wavelet toolbox and Blowfish algorithm for enhanced security in C# using Visual Studio 2010.
Following is my devised method:

Step 1: image histogram test
Input: Cover image
Output: Cover image
Action: If the cover image passes the histogram test
Then
accept the cover image
Else
search for another cover image.
End
Step 2: image corrections
Input: Cover image
Output: Processed cover image
Action:
For each pixel of cover image apply level correction end
For each pixel of cover image apply contrast correction
end
For each pixel of cover image apply color balance correction
end
End

Step 3: Apply wavelet transform
Input: Processed cover image
Output: Transformed image
Action: Apply 2D wavelet transform
End

Step 4: Calculate threshold (Th)
Input: Transformed image
Output: number of pixels can be changed (n), DWT of the cover image
Action: Calculate Th
Th = (correlation factor/Number of coefficient) $\sum$ (mod value of all 2D wavelet coefficients) (1)
Select DWT coefficient of every pixel in cover image
if coefficient value < Th
then
keep coefficient index and n=n+1
end
End

Step 5: Message encryption process
Input: Information
Output: encrypted information
Action: Pass control from Matlab 7.9 to Visual Studio 2010
Encrypt information with Blowfish
Pass control back to Matlab
End

Step 6: Bit streaming
Input: n, encrypted information
Output: 1D bit streamed information
Action: convert the encrypted information to 1D bit stream using n
End

Step 7: 2D wavelet transform of encrypted information
Input: encrypted bit stream of the message
Output: DWT transform of the encrypted message.
Action: transform the encrypted bit stream of the message to Wavelet domain
End

Step 8: Coefficient placement
Input: 2D wavelet transform of cover image and 2D wavelet transform of the encrypted information.
Output: stego image
Action:
a. Put coefficients of wavelet transform of encrypted information in the indexes stored in Step 4
b. Perform inverse wavelet transform
End

EXPERIMENTAL RESULTS

We selected three (03) image sizes and created groups each consists of ten (10) jpeg images to perform our testing:

a. 256x256
b. 306x648
c. 512x512

The image imperceptibility is shown by PSNR values [3].

$$MSE = \frac{1}{MN}\sum_{i=0}^{M}\sum_{j=0}^{N}(Xij - \overline{Xij}) \quad (2)$$

$$PSNR\ (dB) = 10\log_{10}\left(\frac{I^2}{MSE}\right) \quad (3)$$

$\overline{X_{ij}}$ represents the i^{th} rows and j^{th} columns of transformed image and X_{ij} represents the i^{th} rows and j^{th} columns of original image. Higher the PSNR value means more difficult to perceive that any hidden message is hidden. In our experimental work we have found that by increasing the payload, PSNR value drops down. We have to make a compromise between payload and PSNR. Table 1 shows the average results for each group of colored images and Figure 2 shows the results in graph to help study results more easily. If we change our set of images, results will change accordingly. The payload values can also be changed by using a different information or content to be stegan by the proposed method.

As it is clear when we increase the value of correlation factor, value of MSE increases and ultimately decreases the value of PSNR (see Table 1). It was experimentally tested that our method is an improved one as it is less visible to others that any information is hidden in image that makes is more secure from enemies attacks. We have shown a 512x512 image and its stego image in Figure 3a and 3b. In our proposed method, we provided with the double security. Information is encrypted and bit streamed before performing wavelet transform on information. Then wavelet transform of information is placed in wavelet domain of cover image.

CONCLUSIONS

In this paper we presented an improved image steganography technique for any colored image of any size using wavelet transform and blowfish encryption algorithm in comparison with the work presented in [8]. This new method gives better invisibility and security of communication. Our method provides double security by involving blowfish, which satisfies the need of imperceptibility. Future work may be carried out to increase the payload and maintain the higher PSNR values.

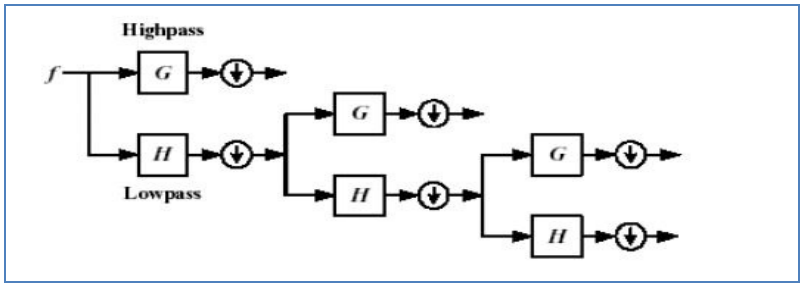


Fig 1: 2DWT

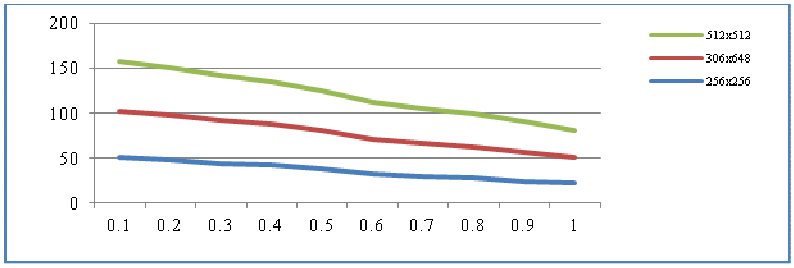


Fig 2: If Comparison of results for three sets of images

Fig 3a : Cover Image

Fig 3b : Stego Image

Table 2: Average Payload and PSNR values for 256x256, 306x648 and 512x512 images

Image Size	Correlation Factor	Payload %	MSE %	PSNR dB
256x256	.1	31.544	0.3241	50.286
	.2	40.094	0.453	47.207
	.3	45.069	0.61	43.61
	.4	51.226	1.544	41.677
	.5	55.545	1.835	38.122
	.6	62.933	2.673	32.659
	.7	64.675	3.588	29.926
	.8	67.501	4.837	27.754
	.9	69.258	6.142	24.257
	1	72.126	7.906	21.984
306x648	.1	38.395	0.349	52.046
	.2	42.703	0.543	50.079
	.3	49.946	0.693	48.284
	.4	56.858	1.2536	45.799
	.5	61.938	1.936	42.284
	.6	65.853	2.775	38.348
	.7	71.038	3.766	36.823
	.8	71.621	4.873	34.563
	.9	73.606	6.38	32.363
	1	74.684	7.824	28.233
512x512	.1	39.275	0.3931	55.251
	.2	44.176	0.552	52.511
	.3	50.389	0.754	50.594
	.4	56.544	1.353	47.844
	.5	61.432	1.974	44.159
	.6	65.886	2.826	41.271
	.7	69.412	3.824	38.566
	.8	71.784	5.031	36.525
	.9	74.164	6.366	34.115
	1	76.143	7.671	31.065

REFERENCES

[1] N. Provos, P. Honeyman, "Hide and seek: An introduction to steganography", IEEE Security Privacy Magazine (2003), volume: 1, Issue: 3, Publisher: IEEE Security & Privacy, Pages: 32-44

[2] Johnson, N.F. & Jajodia, S., "Exploring teganography: Seeing the Unseen", Computer Journal, February 1998

[3] Almohammad, A.; Ghinea, G.; "Stego image quality and the reliability of PSNR", 2nd International Conference on Image Processing Theory Tools and Applications (IPTA), 2010, Pages: 215 – 220

[4] Ali Al-Ataby and Fawzi Al-Naima, "A Modified High Capacity Image Steganography Technique Based on Wavelet Transform", The International Arab Journal of Information Technology, Vol. 7, No. 4, October 2010, Pages: 358-364

[5] Aamer Nadeem et al, "A Performance Comparison of Data Encryption Algorithms", IEEE 2005

[6] http://scholar.lib.vt.edu/theses/available/etd-2062002-152858/unrestricted/Chapter4.pdf

[7] Amitava Nag, Sushanta Biswas, Debasree Sarkar & Partha Pratim Sarkar, "A Novel Technique for Image Steganography Based on DWT and Huffman Encoding", International Journal of Computer Science and Security, (IJCSS), Volume (4): Issue (6)

[8] Ali Al-Ataby and Fawzi Al-Naima, "A Modified High Capacity Image Steganography Technique Based on Wavelet Transform", The International Arab Journal of Information Technology, Vol. 7, No. 4, October 2010

[9] M. F. Tolba, M. A. Ghonemy, I. A. Taha, and A. S. Khalifa, "Using Integer Wavelet Transforms in Colored Image-Steganography", *IJICIS Vol. 4 No. 2, July 2004*

References

[1] http://www.tech-faq.com/steganography.shtml

[2] http://wiki.answers.com/Q/What_is_the_difference_between_stegenography_and_watermarking

[3] http://en.wikipedia.org/wiki/Security_through_obscurity

[4] http://www.jjtc.com/stegdoc/sec202.html

[5] http://itslab.csce.kyushu-u.ac.jp/sakurailab/research/fingerprint.html

[6] http://en.wikipedia.org/wiki/Digital_signature

[7] http://www.scribd.com/doc/20529/Seminar-on-Steganography?autodown=doc

[8] Petit colas, Fabien A.P., "Information Hiding: Techniques for Steganography and Digital Watermarking", 2000.

[9] Sellars, D., "An Introduction to Steganography", URL: http://www.cs.uct.ac.za/courses/CS400W/NIS/papers99/dwellers/stego.html

[10]: http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA349102

[11] W. Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for data hiding", *IBM Systems Journal*, vol. 35, Issues 3&4, 1996, pp. 313-336.

[12] D. Huang and H. Yan, "Interword Distance Changes Represented by Sine Waves for Watermarking Text Images", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 12, December 2001, pp.1237-1245

[13] K. Rabah, "Steganography-The Art of Hiding Data", *Information Technology Journal*, vol. 3, Issue 3, 2004, pp. 245-269.

[14] S.H. Low, N.F. Maxemchuk, J.T. Brassil, and L. O'Gorman, "Document marking and identification using both line and word shifting", *Proceedings of the Fourteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '95)*, vol.2, 2-6 April 1995, pp. 853 - 860.

[15] Y. Kim, K. Moon, and I. Oh, "A Text Watermarking Algorithm based on Word Classification and Inter-word Space Statistics", *Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR'03)*, 2003, pp. 775–779.

[16] S.H. Low, N.F. Maxemchuk, J.T. Brassil, and L.O'Gorman, "Document marking and identification using both line and word shifting", *Proceedings of the Fourteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '95), vol.2, 2-6 April 1995, pp. 853 - 860.*

[17] M. Kharrazi, H. T. Sencar, N. Memon, Image Steganography: Concepts and Practice, *Lecture Note Series, Institute for Mathematical sciences, National University of Singapore, 2004.*

[18] "Issues in Information Hiding Transform Techniques," NRL/MR/5540-02-8621 http://chacs.nrl.navy.mil/publications/CHACS/2002/2002chang-NRL-MR-5540-02-8621.pdf #search='2D%20DFT%20in%20steganography

[19] W. Pennebaker and J. Mitchell. "JPEG STILL IMAGE DATA COMPRESSION STANDARD". van Nostrand Reinhold, 1993.

[20] L. Marvel "Image Steganography for Hidden Communication". *Ph.D. Dissertation, Univ. of Delaware, Dept of EE, 1999.*

[21]. Robert T. McKeon "Strange Fourier Steganography in Movies", *Proceedings of IEEE EIT 2007*

[22] Mohammad Shirali-Shahreza, M. Hassan Shirali-Shahreza "Text Steganography in SMS", *International Conference on Convergence Information Technology 2007*

[23]: H. Motameni, M. Norouzi, M. Jahandar, and A. Hatami "Labeling Method in Steganography" *PROCEEDINGS OF WORLD ACADEMY OF SCIENCE, ENGINEERING AND TECHNOLOGY VOLUME 24 OCTOBER 2007 ISSN 1307-6884*

[24]: M. Chaumont and W. Puech, " A Color Image Hidden in a Grey-Level Image" *Laboratory LIRMM, UMR CNRS 5506, University of Montpellier II 161, rue Ada, 34392 MONTPELLIER CEDEX 05, FRANCE*

[25]: Elvin M. Pastorfide and Giovanni A. Flores, " An Image Steganography Algorithm for 24-bit Color Images Using Edge-Detection Filter ", *INSTITUTE OF COMPUTER SCIENCE, CMSC 190 SPECIAL PROBLEM*

[26]: Mohammad Tanvir Parvez and Adnan Abdul-Aziz **"**RGB Intensity Based Variable-Bits Image Steganography**"** *Gutub College of Computer Sciences & Engineering King Fahd University of Petroleum & Minerals, Dhahran 31261, Saudi Arabia*

[27] Wen-Yuan Chen, "Color image steganography scheme using DFT, SPIHT codec, and modified differential phase-shift keying techniques**",** *Department of Electronic Engineering, National Chin-Yi University of Technology, 35 Lane 215, Section 1, Chung-Shan Road, Taiping City, Taichung County 411, Taiwan, ROC, 2007*

[28]: Ran-ZanWang∗,Yao-De Tsai, "An image-hiding method with high hiding capacity based on best-block matching and *k*-means clustering", *Pattern Recognition Society. Published by Elsevier 2006176464*

[29] National Bureau of Standards - Data Encryption Standard, FIPS Publication 46, 1977.

[30] M Hellman "DES will be totally insecure within ten years" IEEE Spectrum 16(7), Jul 1979, pp 31-41.

[31] Mitsuru Matsui, "Linear Cryptanalysis Method for DES Cipher", in *Advances in Cryptology - Eurocrypt'93*, Lecture Notes in Computer Science, Vol 765, Springer-Verlag, pp 386-397, 1993.

[32] N. Provos and P. Honeyman. "Hide and Seek: An Introduction to Steganography", IEEE: Security & Privacy, vol. 1, pp. 32-44, 2003.

[33] Electronic Frontier Foundation, "Cracking DES", O'Reilly, 1998. 1-56592-520-3. http://www.oreilly.com/catalog/crackdes/

[34] W. Stallings, "Cryptography and Network Security - Principles and Practice", 3rd Edition, Prentice-Hall, 1998. http://www.shore.net/~ws/Security.html

[35] R. Gonzalez and R. Woods, "Digital Image Processing", Sec. Edition. pp 373-374.

[36] S. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation", IEEE Pattern Anal. and Machine Intell., vol. 11, No. 7, pp 674-693, 1989.

[37] M. Do and M. Vetterli, "The contourlet transform: An efficient directional multiresolution image representation," *IEEE Transactions on Image Processing*, vol. 14, no. 12, pp. 2091–2106, Dec. 2005.

[38] M. V. V. Velisavljevic, B. Beferull-Lozano and P. Dragotti, "Directionlets: anisotropic multi-directional representation with separable filtering," *IEEE Transactions on Image Processing*, vol. 15, no. 7, pp. 1916 – 1933, Jul. 2006

[39] E. J. Cand`es and D. L. Donoho, "Curvelets-a surprisingly effective nonadaptive representation for objects with edges," *Curve and Surface Fitting*, 1999.

[40] E. L. Pennec and S. Mallat, "Sparse geometric image representation with bandelets," *IEEE Transactions on Image Processing*, vol. 14, no. 4, pp. 423–438, Apr. 2005.

[41] S.Mallat and F.Falzon, "Analysis of low bit rate image transform coding," *IEEE Transactions on Signal Processing*, vol. 46, no. 4, pp. 1027–1042, Apr. 1998.

[42] P. J. Burt and E. H. Adelson, "The laplacian pyramid as a compact image code," *IEEE Transactions on Communications*, vol. 31, pp. 532–540, Apr. 1983.

[43] M. Do and M. Vetterli, "Framing Pyramid", *IEEE Transactions on Signal Processing*, vol. 51, no. 9, pp. 2329–2342, Sept. 2003.

[44] M. N. Do, "Contourlets and sparse image expansions," Ph.D. dissertation, Department of Electrical and Computer Engineering University of Illinois, Urbana IL, 2003

[45] R.H.Bamberger andM.J.T.Smith, "Directional decomposition of images: Theory and design," *IEEE Transactions on Signal Processing*, vol. 4, no. 4, pp. 882–893, Apr. 1992

[46] J. Fridrich and M. Goljan, \Digital image steganography using stochastic modulation," *SPIE Symposium on Electronic Imaging, San Jose, CA,*, 2003.

[47] Nameer N. EL-Emam, "Hiding a large amount of data with high security using steganography algorithm", Journal of Computer Science, Page(s): 223 – 232, April 2007.

[48] S.K.Bandyopadhyay, Debnath Bhattacharyya, Poulumi Das, S. Mukherjee, D. Ganguly, "A Secure Scheme for Image Transformation", IEEE SNPD, pp. 490-493, August 2008.

[49] Adel Almohammad and Gheorghita Ghinea, “Image Steganography and Chrominance Components”, 2010 10th IEEE International Conference on Computer and Information Technology (CIT 2010), pp 996-1001

[50] Mrs. Gyankamal J. Chhajed Mrs. Vandana Inamdar Mrs Vahida Attar, “Steganography in Black and White Picture Images”, 2008 Congress on Image and Signal Processing, pp 141-144

[51] NIST, "Advanced Encryption Standard Call", NIST, 1997. http://www.nist.gov/aes/

[52] NIST Advanced Encryption Standard (AES) Development Effort web site http://csrs.nist.gov/encryption/aes/aes -home.htm

[53] Joan Daemen, Vincent Rijmen, "AES Proposal: Rijndael", Banksys/Katholieke Universiteit Leuven, Belgium, AES submission, Jun 1998. http://www.esat.kuleuven.ac.be/~rijmen/rijndael/

[54] R. Rivest, A. Shamir, and L. Adleman, “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems”, Communications of the ACM 21,2 (Feb. 1978)

[55] Juan José Roque, Jesús María Minguet, “SLSB: Improving the Steganographic Algorithm LSB”, http://www.fing.edu.uy/inco/eventos/cibsi09/docs/Papers/CIBSI-Dia3-Sesion9(1).pdf

[56] Hedieh SAJEDI, Mansour JAMZAD, “ ContSteg: Contourlet-Based Steganography Method”, *Wireless Sensor Network*, 2009, 3, 163-17

[57] M. F. Tolba, M. A. Ghonemy, I. A. Taha, and A. S. Khalifa, “USING INTEGER WAVELET TRANSFORMS IN COLORED IMAGE-STEGANOGRAPHY”, *IJICIS Vol. 4 No. 2, July 2004*

[58] K B Shiva Kumar, K B Raja, R K Chhotaray, Sabyasachi Pattanaik, “BIT LENGTH REPLACEMENT STEGANOGRAPHY BASED ON DCT COEFFICIENTS”, International Journal of Engineering Science and Technology Vol. 2(8), 2010, 3561-3570

[59] Amitava Nag, Sushanta Biswas, Debasree Sarkar & Partha Pratim Sarkar, “A Novel Technique for Image Steganography Based on DWT and Huffman Encoding”, International Journal of Computer Science and Security, (IJCSS), Volume (4): Issue (6)

[60] Bo Yang and Beixing Deng, “Steganography in Gray Images Using Wavelet” ISCCSP 2006

[61] Hedieh Sajedi · Mansour Jamzad, Using contourlet transform and cover selection for secure steganography Published online: 5 August 2010 © Springer-Verlag 2010

[62] Ali Al-Ataby1 and Fawzi Al-Naima2, “A Modified High Capacity Image Steganography Technique Based on Wavelet Transform”, The International Arab Journal of Information Technology, Vol. 7, No. 4, October 2010

[63] Aamer Nadeem,M Y Javed “A Performance Comparison of Data Encryption Algorithms”. IEEE 2005