Republic of Iraq
Ministry of Higher Education and Scientific Research
Al-Nahrain University
College of Science

Low Rate Hiding in Audio Data Using Phase Domain

A Thesis

Submitted to College of Science of AL-Nahrain University in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Science

By

Hiba Mohammed Saleem AL-Kawaz

(B.Sc. 2003)

Supervisor

Dr. Loay E. George

February 2007 *Safar 1428*

بسم الله الرحمن الرحيم

اقْرَأْ وَرَبُّكَ الْأَكْرَمُ ﴿٣﴾ الَّذِي عَلَّمَ بِالْقَلَمِ ﴿٤﴾
عَلَّمَ الْإِنسَانَ مَا لَمْ يَعْلَمْ ﴿٥﴾

صدق الله العظيم

سورة العلق

Supervisor Certification

I certify that this thesis was prepared under my supervision at the Department of Computer Science / College of Science/ Al-Nahrain University, by **Hiba Mohammed Saleem AL-Kawaz** As partial fulfillment of the requirement for the degree of Master of Science in Computer Science.

Signature:
Name: **Dr. Loay E. George**
Title: **Assistan Professor**
Date: **/ / 2006**

In view of the available recommendations, I forward this thesis for debate by the examination committee.

Signature:
Name : **Dr. Taha S. Bashaga**
Title : **Head of the department of Computer Science,**
Al-Nahrain University.
Date : / / **2006**

Certification of the Examination Committee

We chairman and members of the examination committee, certify that we have studies this thesis "**Low Rate Hiding in Audio Data Using Phase Domain**" presented by the student **Hiba Mohammed Saleem AL-Kawaz** and examined her in its contents and that we have found it worthy to be accepted for the degree of Master of Science in Computer Science with excellent degree.

Signature:

Name: **Dr. Ayad A. AL-Ani**
Title : **Assistant Professor**
Date : / /**2007**
(Chairman)

Signature:

Name: **Dr. Bushra K. AL-Abudi**
Title : **Assistant Professor**
Date : / /**2007**
(Member)

Signature:

Name: **Dr. Ban N. Thanoon**
Title : **Lecturer**
Date : / /**2007**
(Member)

Signature:

Name: **Dr. Loay E. George**

Title : **Assistant Professor**

Date : / /**2007**

(Supervisor)

Approved by the Dean of the Collage of Science, Al-Nahrain University.

Signature:

Name: **Dr. LAITH ABDUL AZIZ AL-ANI**
Title : **Assist. Prof**.
Date : / /**2007**
(Dean of Collage of Science)

Dedication

To my father and mother

With my love

Acknowledgment

First of all great thanks are due to Allah who helped me and gave me the ability to achieve this research from first to last step.

I would like to express my deep gratitude and sincere thanks to my supervisor ***Dr. Loay E. George*** *for guidance, assistance and encouragement during the course of this project.*

Grateful thanks for the Head of Department of Computer Science of Al-Nahrain University ***Dr. Taha S. Bashagha*** *for the continuous support during the period of my studies.*

Deep gratitude and special thanks to my family: ***my parent, brothers, and sisters*** *for their encouragements and supporting to succeed in doing this work.*

Deep thanks to my uncle ***Dr. Sabah S. AL-Kawaz*** *for his support, interest and generosity.*

Special thanks to all my friends for giving me advises.

Hiba

Abstract

Steganography is considered one of the widely used methods for hiding information; it hides secret data in digital cover without clear suspicion.

This work focus on studying two methods for hiding any secret data type in the audio signal. First, the audio signal is transformed to frequency domain by using quick Fourier transform that depended on the reduction of time and number of mathematical operations.

Two hiding methods are designed to embed secret data bits in the phase domain coefficients of audio signal. The first method, called hiding in voiced blocks (HVB), implies the inserting of secret bits in the voiced parts of audio data. While the second method, called hiding in checked blocks (HCB), implies the insertion of secret bits in audio blocks that successfully passed the integrity retrieved bits tests. These two methods use two types of wave audio cover files channels (mono and stereo).

Finally, the performance of the proposed hiding methods was tested by using fidelity measures mean square error (MSE), signal noise ratio (SNR), and peak signal noise ratio (PSNR) to measure the rate of error, and the hiding rate was computed to assess the embedding power of each hiding method. Also, the effects of some control parameters on the system performance were investigated to assist user to correctly choose the values of system parameters.

The test results indicate that the HVB method shows more embedding power than HCB method, but not all the embedded secret bits could correctly retrieved when HVB method is utilized.

List of Abbreviations

ADC	Analog to Digital Converter
DAC	Digital to Analog Converter
dB	Decibel
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
FAT	File Allocation Table
FFT	Fast Fourier Transform
HAS	Human Audio System
HCB	Hiding in Checked Blocks
HTML	Hiber Text Mark Language
HVB	Hiding in Voiced Blocks
HVS	Human Visual System
Hz	Hertz
IDFT	Inverse Discret Fourier Transform
KB	Kilo Byte
KHz	Kilo Hertz
LBE	Low Bit Encoding
LSE	Least significant Bit
MB	Mega Byte
MSE	Mean Square Error
OSI	Open System Interconnect
PCM	Pulse Code Modulation
PSNR	Peak Signal to Noise Ratio
QFT	Quick Fourier Transform
RIFF	Resource Interchange File Format
SNR	Signal to Noise Ratio
TCP/ IP	Transmission Control Protocol / Internet Protocol
WAV	Windows Audio Visual

List of Symbols

Symbol	Decryption
C	Vector contain cosine kernel values
Cv	Cover medium without embedded message
Cv'	Cover medium with embedded message
D	Decode process
E	Encode process
e	Uniform random signal
F	Vector of transform frequency domine
f	Vector of time domain
F_i	Vector of Imaginary part coefficients
F_r	Vector of Real part coefficient
F^{-1}	Vector of inverse transform
fn	Float number
HR	Hiding ratio
I	Integer number
i	Counter
j	Complex number $\sqrt{-1}$
K	Stego key
M	Total number of audio samples
Mag	Vector of magnitude coefficients
Msg	Embedded message
N	Audio block size
OverInfo	Vector of overhead bits
Phs_0	Absolute phase
Phs	Vector of phase coefficients
Phs_H	Vector of host phase coefficients
Phs_q	Vector of quantized phase values

Q	Quantization step
q	Unquantized signal
$q^{'}$	Quantized signal
R	Slack step ratio
S	Vector contain sine kernel values
$Scrt$	Vector of secret bits
Sg_0	Segment zero
Sg_n	Segment n
s_i	i^{th} sample in the audio cover data
s_i'	i^{th} sample in the audio stego-cover data
S_t	Total number of secret bits insertion
S_w	number of wrong extracted secret bits
Thr	Vector contain cosine values
u	Frequency counter
WR	Wrong retrieved secret bits ratio
x	Time counter
α	The bounding area
β	Shifting ratio
Δ	Quantized adding ratio

Contents

Chapter Three (The Proposed System Implementation)

Chapter Four (Performance Test Results)

Chapter Five (Conclusion and Suggestions)

References

Chapter One
General Introduction

1.1 Preface

When humans borne, special and secret things and information are borne with them. Some of these secret information needs to be transmitted between them. These needs stimulate the question about how secret information can be transmitted between people without discover. Also, when wars happened in the world, the secret strategic data had increased, and the techniques to protect these data are developed. Nowadays, two kinds of techniques were used to conceal information, they are: Encryption or information hiding. The importance of these two protection methods was vastly increased since the discovery of telephone, fax, electronic communication and computer. Also its importance was exploded when the internet enter people lives and become the best, fast and common way to communication in the world. Internet provides the facilities to exchange text, image, audio, and video between users, and its access reach sensitive locations (like, martial locations, political locations) for each government in the world, also it is the most public way to link with large companies and banks in world. All these facts encouraged some persons (or companies) to developed ways to steal information, and to get some tools (software) to make un-authorized access to closed locations.

Still, it is critical problem to secure multimedia contents against illegal use, such digital contents can be easily copied or edited on a computer and delivered through the network by a third party without permission of the copyright owner. In order to solve this problem, data hiding has got great attention as a promising method that plays a complementary role to the conventional cryptographic techniques.

1.2 Information Hiding

Sometimes it is better hiding messages rather than enciphering them. In fact, the main purpose of cryptography is to make message incomprehensible, so that people, who do not posses secret keys, cannot recover the message. Instead, the data hiding uses binary files with certain degree of irrelevancy and redundancy to hide data. Digital books, images, videos, and audio tracks are ideal for this purpose. Digital representation of signals brings many advantages when compared to analog representation, and these advantages are [CF00]:

1. Lossless recording and copying.
2. Convenient distribution over network.
3. Easy editing and modification.
4. Easily searchable archival.
5. Durable.
6. Cheap.

Against these advantages some serious problems were appeared:

1. Wide spread copyright violation.
2. Illegal copying and distribution.
3. Problematic authentication.
4. Easy forging.

The general definitions of hiding data in other data can be described as follows: the *embedded data* is the message that a person wishes to send secretly. This message must be concealed in a normal message as a *cover-text*, or *cover-image*, or *cover-audio*, or in general a *cover-object*, producing the *stego-object* or the *marked-object*. In particular, a *stego-key* is necessary to control the hiding process, to restrict detection and recovery of the embedded data to un-authorized people. The hidden data may have

no relationship with (or may provide important information about) the cover-object, in which it is, embedded [CF00]. The classification of data hiding techniques is presented in figure (1.1).

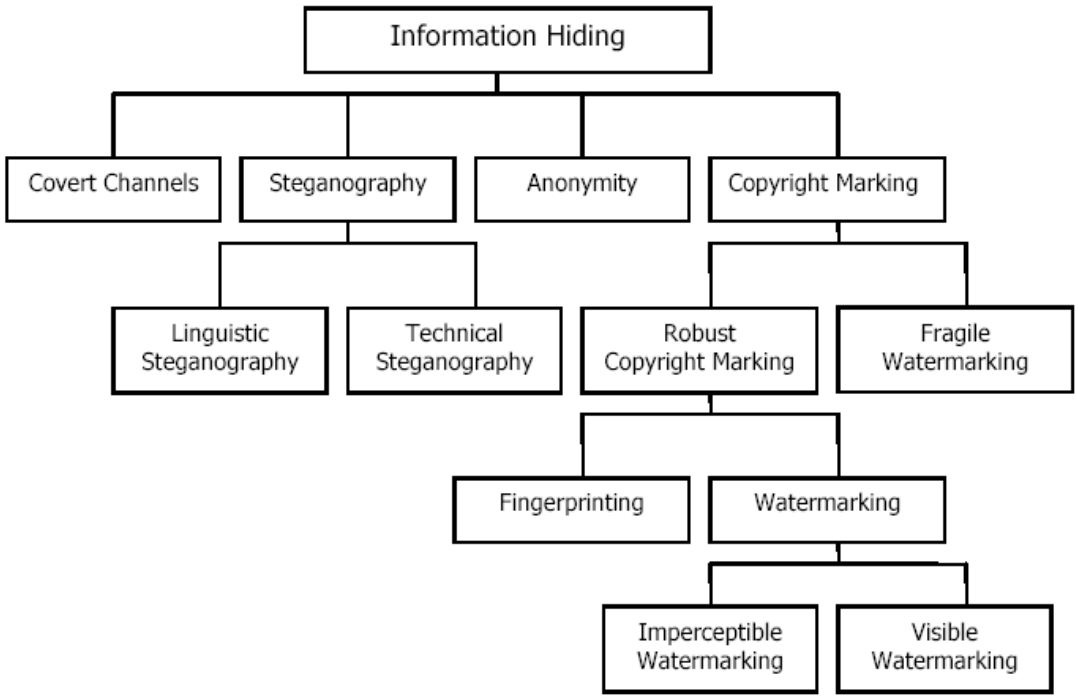


Fig (1.1) A classification of information hiding techniques [CF00]

Data-hiding techniques should be capable of embedding data in a host signal with the following restrictions and features [Ben96]:

1. The host signal should be nonobjectionally degraded, and the embedded data should be minimally perceptible. This means that the observer should not notice the presence of the data, even if it is perceptible.
2. The embedded data should be directly encoded into the media, rather than into a header, so that the data remain intact across varying data file formats.
3. The embedded data should be immune to modifications ranging from intentional and intelligent removal attempts, to anticipated

manipulations, e.g., channel noise, resampling, encoding, lossy compressing, digital-to-analog (D/A) conversion, etc.

4. Asymmetrical coding of the embedded data is desirable, since the purpose of data hiding is to keep the data in the host signal, but not necessarily to make the data difficult to access.
5. Error correction coding should be used to ensure data integrity. It is inevitable that there will be some degradation in the embedded data when the host signal is modified.
6. The embedded data should be self-clocking or arbitrarily re-entrant. This ensures that the embedded data can be recovered when only fragments of the host signal are available.

1.3 Steganography

Steganography is one of the classes (applications) of information hidings. The word steganography is hard to find in any dictionary. It comes from the Greek word "steganos" (means *covered*) and the "graphy" (means *writing*), so steganography literally means "covered writing". This aims to transmit a message through a channel where some other kinds of information are already being transmitted [PWJ99]. The general principle of steganography is illustrated in figure (1.2), from this figure the following definitions are needed to understand the involved components of any steganography system.

1. Cover medium: is the host medium in which the secret data is hidden, it can be an innocent looking piece of information, or some important media that must be protected against copyright or integrity reasons [JJ98-1]. Covers are supposed to contain data that is uninteresting to the enemy and will be unlikely to be subject to any sort of analysis.

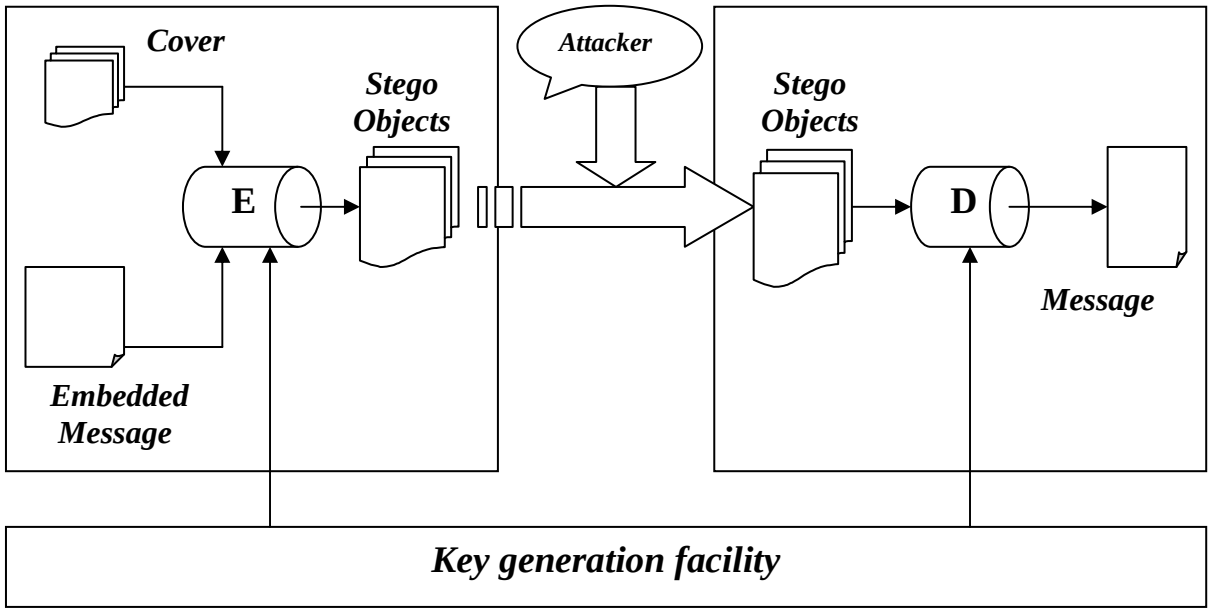


Fig (1.2) General scheme of steganography, E is encode part and D is decode part [KP00]

In addition, cover should never used twice, since an attacker who has access to the two versions of the cover can easily detect and reconstruct the hidden message. To avoid accidental reuse, both sender and receiver should destroy all covers they have already used for information transfer [KP00].

***2. Embedded message*:** is the hidden message that wants to be put in the cover. It could be some data for steganography, copyright information, or some added contents for digital watermarking [JJ98-1].

3. Stegokey: It is represented by some secret information, which is needed in order to extract the embedded message from the stego-object. A stegokey is a bit of extra information that the receiver must know in order to recover the message from the container. The use of a stego key is desirable for two reasons; (1) it may make hidden data statistically harder to detect, and (2) the stego key frustrate unauthorized attempts to obtain the hidden message from the cover [JJ98-1].

4. Stego object: is the output part produced from steganography engine. Steganography could be expressed as follows [JJ98-1]:

Cover medium + Embedded message = Stego object

When the *stego object* is produced then the sender transmits it over an insecure channel to receiver. Receiver can reconstruct *secret message* from *stego object*, since he know the embedding method and stego-key that used in the embedding process.

In a perfect steganography system, the original cover should not be distinguishable from a stego object, neither by human nor by a computer looking for statistical pattern.

Unfortunately, some problems may hinder transmission *stego object* in insecure channel. The attackers could be classified as *passive attacker* and *active attacker* or even malicious attacker [KP00]. There are a critical distinction between *passive attacker* (who can monitor traffic and signal to some process outside the system if unauthorized traffic is detected) and *active warden* (who try to remove all possible covert messages from traffic that passes through his hands, or even worse when he can act in a *malicious* way, he can send new message instead the original stego message to receiver) [AP98].

1.4 Related Work

Several researches concerned with information hiding in audio techniques have been developed. Some of the developing previous works related to the thesis objective are given below:

1. Petrovic, with his colleagues, (1999) [PWJ99] present data hiding within audio signal using a proposed approach called (the short-term autocorrelation modulation). In this method the difference signal is added to the cover signal to obtain a stego signal. This difference signal depends on the cover signal, embedded data as well as the stego key.

2. Polpitiya and Khan (2001) [PK01] proposed a technique for hiding message in the audio file. *Least Significant Bit* (LSB) technique was used as a hiding method. To increase the level of security the message was encrypted before insertion in audio file.

3. Yasmeen (2003) [Yas03] presented a hiding system that embeds a 10 characters digital watermark in the audio signal. The system uses two different hiding techniques: *low bit encoding* (LBE) in the time domain and the human auditory characteristics in the frequency domain by using *Fourier Transform*. She noticed that the best locations to host secret data in the spectrum are in the middle frequency coefficients of each audio frame, and no more than one bit could be embedded in each frame. Therefore, the number of embedded bits was less than that could be attained by using embedded methods in the time domain.

4. Witwit (2004) [Wit04] designed an audio in audio steganography system that hide data of wave file of 8-bits sampling resolution and single channel into cover wave file of 8-bits sampling resolution. He applied wavelet transform on both secret and cover data. To obtain high and low coefficient, then he made suitable gaps (by using quantization process) for inserting the secret data in the cover coefficients. After that a down-scale process was implemented to minimize the scope of the quantized coefficients. In the embedding stage, he embedded the downscaled coefficients of the secret audio in the cover detail (high) coefficients, and the overhead information (i.e., size of secret data, quantization steps) was embedded in the low coefficients of the cover.

5. Majeed (2004) [Maj04] introduced a steganography system that hides audio in audio using *Discrete Cosine Transform* (DCT). He suggested six hiding methods; some of them dedicated to hide audio

data in audio data, while others have the ability of hiding any type of secret data in audio data. Also, he applied some additional steps to increase the security immunity of the system by using two kinds of encryption: *pure key steganography* and *private key steganography*.

6. Muntaha (2004) [Mun04] introduced an image in image steganography system. She utilized *Fourier Transform Coding* to hide the desired information data. The hiding process was implemented on the phase component in Fourier domain by using color correspondence.

1.5 Aim of Thesis

The aims of this work to design and implement steganography system that hides secret data in audio data by using Fourier transform domain. The secret data is a binary file, whatever it is (image, audio, text,…). The embedding method is done on the phase component of the cover audio signal that produced after applying Fourier Transform. During the embedding stage, the symmetry attribute of the Fourier domain should be taken into consideration, to keep the validity of the reconstructed cover data. In order to minimize the error rate that may occur in the embedded secret data, some additional steps are implemented, they based on making statistical tests on the extracted repeated signal, or by avoiding hiding in the weak parts of the cover audio signal.

1.6 Thesis Layout

In addition to chapter one, this thesis consists of the following chapters:

Chapter Two: (Theoretical Background)

In this chapter some steganography techniques and steganalysis concepts are presented. Also, the discrete Fourier transform theory and the concept of sound signal are explained.

Chapter Three: (The Proposed System Implementation)

In this chapter, the proposed system design steps and the implementation steps are given. The hiding module with two hiding methods, and extraction modules are presented.

Chapter Four: (Performance Test Results)

This chapter is dedicated to present the results of the tests that performed on the system by using different types of audio samples.

Chapter Five: (Conclusions and Suggestions)

Some conclusion remarks that derive from the results of the conducted tests are given in this chapter; also, some suggestions for future work are listed.

Chapter Two
Theoretical Background

2.1 Introduction

This chapter introduces the various techniques of steganography, which depend on the type of cover media. Also the methods used to hide information in audio data are reviewed. The possible attacks *"steganalysis"* may impose on steganography systems and the relevant security are preview. When dealing with the sound we might need a way to transform the audio from time domain to frequency domain, Fourier transform is one of the most commonly used transforms and it will be used in this research to accomplish hiding task. Therefore the aspects and equations deal with Fourier transform are described. The basic structure of sound and the way to represent in digital form is also given in this chapter.

2.2 Steganography Techniques

The internet provides an increasing broad band of communication service as a means to distribute information between the users. Such information includes text, image and audio…etc. Such kind of distributed information provides excellent carriers for hidden information. Many different techniques have been introduced to utilize these carriers as hosts for hiding information, also some other kind of carriers where used (include storage device and transmission control protocol / internet protocol (TCP\IP) packet) [JJ98-2]. According to the type of cover media the steganography techniques are classified to the following:

2.2.1 Hiding in Image [Ben96]

Data hiding in still images presents a variety of challenges that arise due to the way of the human visual system (HVS) works and the typical modifications that images undergo. There are many attributes of the HVS that make images potential candidates for exploitation in a data-hiding system:

1. The masking effect of edges.
2. The varying sensitivity to contrast as a function of spatial frequency.
3. The low sensitivity to small changes in luminance for random patterns.
4. The low sensitivity to very low spatial frequencies, such as continuous changes in brightness through an image.

One of the most important advantages in using still images for data hiding is that they represent a no causal medium, since it is possible to access any pixel of the image at random. There are various techniques for data hiding in still images:

1. Least significant bit (LSB) insertion.
2. Spread spectrum.
3. Texture block.
4. Patchwork.
5. Orthogonal projection coefficients manipulation.
6. Other methods: dithering manipulation, perceptual masking, DCT coefficients manipulation.

Some techniques are more suited to deal with small amounts of data, while others to large amounts. Some techniques are highly resistant

to geometric modifications, while others are more resistant to nongeometric modifications...etc.

2.2.2 Hiding in Text

An early approach to hiding information is in text. Invisible inks prove to be a popular medium. Computers bring more capability to information hiding. The layout of a document may also reveal information. Documents may be marked identified by modulations in the positions of lines and words. Adding spaces and "invisible" characters to text provides a method to pass hidden information [JJ98-2]. Data hiding in text is an exercise in the discovery of modifications that are not noticed by readers. For example, HTML files can be used to carry information since adding space, tabs, invisible characters, and extra line breaks are ignored by web browsers. There are three major methods of encoding data: open space methods (through the manipulation of white space; unused space on the printed page), syntactic methods (that utilize punctuation) and semantic methods (through the manipulation of the words themselves) [Ben96].

2.2.3 Hiding in Disk Space

There are unused or reserved spaces on the disk that could be used to hold covert information; it provides a means for hiding information without perceptually degrading the carrier [JJ98-2]. The way operating system store files typically results in unused space that appears to be allocated to files. For example, under Windows 95 operating system hard disk (HD) drives are formatted as FAT16 (MS-DOS compatible) without compression they typically use cluster sizes of 32^2 Kilobyte (KB). This means that the minimum space allocated to a file is 32 Kb, if a file is 1

Kb in size, then an additional 31 Kb is "wasted." This "extra" space can be used to hide information without showing up in the directory [KP00].

2.2.4 Hiding in Network Packet [JJ98-2]

Protocol in the Open System Interconnect (OSI) network model has characteristics that can be used to hide information. An uncountable number of data packets are transmitted daily over the Internet, any of which can provide an excellent covert communication channel. For example, TCP packet header has six unused bits and the IP packet header has two reserved bits, they can be used to transport information across the Internet.

2.2.5 Hiding in Video [Lan99]

Video files are generally a collection of images and sounds, so most of the presented techniques on images and audio can be applied to video files too. The great advantages of video are the large amount of data that can be hidden inside and the fact that it is a moving stream of images and sounds. Therefore, any small but otherwise noticeable distortions might go unobserved by humans because of the continuous flow of information.

2.2.6 Hiding in Audio

Embedding secret message in digital sound is generally more difficult than embedding information in digital image because the human auditory system (HAS) is extremely sensitive [KP00]. Sensitivity to additive random noise is also acute. The perturbations in a sound file can be detected as low as one part in ten million (80 dB below ambient level). However, there are some “holes” available. While the HAS has a large dynamic range, it has a small differential range. As a result, loud sounds tend to mask out quiet sounds. Additionally, the HAS is much less

sensitive to the phase components of sound. Finally, there are some environmental distortions so common as to be ignored by the listener in most cases.

The transmission medium of an audio signal refers to the environment in which a signal might go through to reach its destination. Bender and his colleagues categorize the possible transmission environments into the following four groups [Ben96]:

1. Digital end-to-end environment where the sound files are copied directly from one machine to another.
2. Increased/decreased resampling environment where the signal is resampled to a higher or lower sampling rate.
3. Analog transmission and resampling where a signal is converted to an analog state, played on a clean analog line, and resampled.
4. "Over the air" environment where the signal is played into the air and resampled with a microphone.

2.3 Audio Hiding Methods

Different kinds of methods of data hiding in audio were introduced in the published literature, the most common methods are:

2.3.1 Low-Bit Coding

This way is common in steganography and is relatively easy to apply in image and audio [KP00]. By replacing the least significant bit of each sampling point by a coded binary string, we can encode a large amount of data in an audio signal [Ben96]. Also we can change more than one bit of the cover, for instance by storing two message bits in the two least significant bits of one cover element. The secret message should normally have less bits than cover, the embedding process should be finished before recording the end of the cover [KP00].

2.3.2 Phase Coding [Ben96]

The second technique is based on the weakness of HAS insensitivity to initial phase, taking into consideration that HAS is sensitive only for differential phase variation. Thus, the sound file is divided into blocks and each block's initial phase is modified according to the embedded message content while preserving the subsequent relative phase shifts. Phase coding techniques implies the following stages:

1. The original sound sequence is broken into a series of N short segments.
2. Transform each segment into magnitude and phase by using Fourier Transform.
3. Calculate the phase difference between consecutive segments.
4. For segment Sg_0, create an artificial absolute phase Phs_0.
5. For all other segments, create new phase frames.
6. Combine the new phase and original magnitude to get new segment Sg_n.
7. Concatenate the new segments to create the stego output.

Phase coding is one of the most effective coding methods in terms of the signal-to-perceived noise ratio. When the phase relation between each frequency component is, dramatically changed, noticeable phase dispersion will occur. However, as long as the modification of the phase is sufficiently small (sufficiently small depends on the observer; professionals in broadcast radio can detect modifications that are unperceivable to an average observer), an *inaudible* coding can be achieved.

2.3.3 Spread Spectrum [Ben96]

Spread spectrum schemes can be used even if they usually add perceivable noise to the sound. In a normal communication channel, it is often desirable to concentrate the information in as narrow a region of the frequency spectrum as possible in order to conserve available bandwidth and to reduce power.

The basic spread spectrum technique, on the other hand, is designed to encode a stream of information by spreading the encoded data across as much of the frequency spectrum as possible. This allows the signal reception, even if there is interference on some frequencies.

2.3.4 Echo Data Hiding

Echo hiding attempts to hide information in discrete audio signal by introducing an *echo.* The data are hidden by varying three parameters of the echo: initial amplitude, decay rate, and offset (or delay). As the offset (or delay) between the original and the echo decreases, the two signals blend. At a certain point, the human ear cannot distinguish between the two signals. The echo is perceived as added resonance. This point is hard to determine exactly. It depends on the quality of the original recording, the type of sound being echoed, and the listener. In general, it is found that this fusion occurs around 1/1000 of a second for most sounds and most listeners.

The coder uses two delay times, one to represent a binary one (offset) and another to represent a binary zero (offset + delta). Both of these delays were chosen small enough to be heard by the naked ear as enrichments in sound, and not as distortions [Ben96].

All abovementioned methods are considered as low bit coding techniques, they have some sort of robustness. For example, phase coding method provides robustness against resampling of the carrier signal, but

has a very low data transmission rate, since the secret information is encoded only in the first signal segment. On the contrary, spread spectrum and echo hiding perform better in many cases [KP00].

2.4 Steganography Protocols [JDJ00]

There are three types of steganographic protocols *pure steganography, secret key steganography, and public key steganography.* In the following some of their principles will discussed.

2.4.1 Pure Steganography

The steganography system that does not require the prior exchange of some secret information (like a stego key) is called a *pure steganography*. The embedding process can be describe as a mapping $E:Cv\times Msg\rightarrow Cv'$, where *Cv* is the set of the possible covers and *Msg* is the set of possible message. The extraction process consists of a mapping $D:Cv'\rightarrow Msg$, extracting the secret message out of a cover. Both sender and receiver must have access to the embedding and extraction algorithm, but the algorithm should not be public.

2.4.2 Secret Key Steganography

A *secret key steganography* system is similar to a symmetric cipher: the sender chooses a cover *Cv* and embeds the secret message into *Cv* by using the secret key *K*. Receiver knows the key used in the embedding process, he can reverse the process and extract the secret message. Anyone who does not know the secret key should not be able to obtain evidence about the encoded information.

2.4.3 Public Key Steganography

Public key steganography systems require the use of two keys, one private key and one public key; the public key is stored in a public

database. Whereas the public key is used in the embedding process, the secret key is used to reconstruct the secret message. One way to build a public key steganography system is the use of a public key cryptosystem.

2.5 Security of Steganography System [KP00]

When the embedding method doesn't dependent on some secret information shared by sender and receiver, an attack may happen since the receiver is not able to verify the correctness of the sender's identity. To avoid such kind of attack the algorithm must be robust and secure. We can define a *secure steganography algorithm* in terms of four requirements:

1. Messages are hidden using a public algorithm and a secret key, the secret key must identify the sender uniquely.
2. Only the holder of the correct key can detect, extract, and prove the existence of the hidden message. Nobody else should be able to find any statistical evidence of a message's existence.
3. Even if the enemy knows the contents of one hidden message, he should have no chance of detecting others.
4. It is computationally infeasible to detect hidden messages.

2.6 Steganalysis

The steganography deals with techniques for hiding information, while the goal of *steganalysis* is to breaking steganography security. Attacks and analysis on hidden information may take several forms: detection, extraction, and disabling or destroying hidden information. The person who applies this analysis is called the *steganalyst* or *attacker* [JJ98-1].

It is fair to say that steganalysis is both an art and a science. The art of steganalysis plays a major role in the selection of features or

characteristics a typical stego message might exhibit while the science helps in reliably testing the selected features for the presence of hidden information.

In general, extraction of the secret message could be a harder problem than mere detection. Therefore, the attackers may be categorized according to their abilities into:

1. ***Passive steganalysis:*** Detect the presence or absence of a secret message in an observed message.
2. ***Active steganalysis:*** Extract a (possibly approximate) version of the secret message from a stego message.

Note that active steganalysis could be different from an active warden case. An active warden manipulates the stego message in the hopes of destroying the secret message (if any), but an active steganalysis attempts to estimate and extract the secret message without destroying it [Cha02]. The attacks used by steganography can be categorized into [JJ98-1]:

1. **Stego only attack:** Only the stego-object is available to detect and/or extract the embedded message.
2. **Known cover attack:** is used when both the stego-object and the cover object are available, so that the attacker can make comparisons between them.
3. **Known message attack:** such kind of attack implies the analysis of some known patterns that correspond to hidden information, which may help against attacks in the future. Even with small the message, this may be very difficult and may be considered the same as a stego-only attack.
4. **Chosen stego attack:** is used when the stego-object and the steganography tool (or algorithm) are known.

5. **Chosen message attack:** The steganalysis generates a stego-object from some steganography tool or algorithm of a chosen message. The goal of this attack is to determine corresponding patterns in the stego-object that may point to the use of specific steganography tools or algorithms.
6. **Known stego attack:** The steganography tool (algorithm) is known and both the original and stego-object are available.

2.7 Fourier Transform

Fourier analysis is named after **Jean Baptiste Joseph Fourier** (1768-1830), a French mathematician and physicist. While many contributed to the field, Fourier is honored for his mathematical discoveries and insight into the practical usefulness of the related techniques. Fourier was interested in heat propagation, and presented a paper in 1807 to the Institute de France on the use of sinusoids to represent temperature distributions. Fourier analysis is a family of mathematical techniques, all based on decomposing signals into sinusoids (collection of sine and cosine signals) [Smi99].

Time is fundamental in everyday from life. At most all things move as a function of time. On the other hand, although sound waves are composed of moving atoms, their movement is too small and their frequency of the vibration is too fast to observe directly. Thus it is easier to describe sounds in *frequency* space rather than *time* space. The transform of sound or many other things in physics, from *time* space to *frequency* space could be by Fourier transform. In this research work, we will record time-varying sound wave patterns and transform them into frequency spectra (i.e., amplitude as a function of frequency) [Hei05].

The number of samples in the each domain is usually represented by the *variable N*. While *N* can be any positive integer, a power of two is

usually chosen (i.e., 128, 256, 512, 1024, etc). There are two reasons for this. First, digital data storage uses binary addressing this will make power of two is a natural signal length. Second, the most efficient algorithm for calculating the *discrete Fourier transform* (DFT) is the *fast Fourier transform* (FFT), it is usually operates with *N* that is a power of two. In most cases, the samples run from 0 to *N-1*, rather than 1 to *N*.

One of the roots in the Fourier series analysis is the *discrete Fourier transform (DFT)*. It is used with discrete and periodic signal because the signal could be either *continuous* or *discrete*, and it could be either *periodic* or *aperiodic*, figure (2.1) illustrates different type of signals [Smi99]. The discrete Fourier transform is given by [MSY99]:

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) e^{-j 2\pi u x / N} \qquad (2.1)$$

For u=0, 1, 2, ..., N-1

Where, *j* denotes the complex number $\sqrt{-1}$, the great German mathematician Carl Friedrich Gauss (1777-1855) coined this term as **complex number**, and he paved the way for the modern understanding of the field. Every complex number is the sum of two components: a real part and an imaginary part. The real part is a real number, one of the ordinary numbers that learned in childhood. The imaginary part is an imaginary number, that is, the *square-root of a negative number*, and we could express the real and imaginary part as (X + jY), where X real part, and jY is the imaginary part. One of the most important equations in complex mathematics is **Euler's relation**, named for Leonhard Euler (1707-1783)) [Smi99]:

$$e^{jx} = \cos x + j \sin x \qquad (2.2)$$

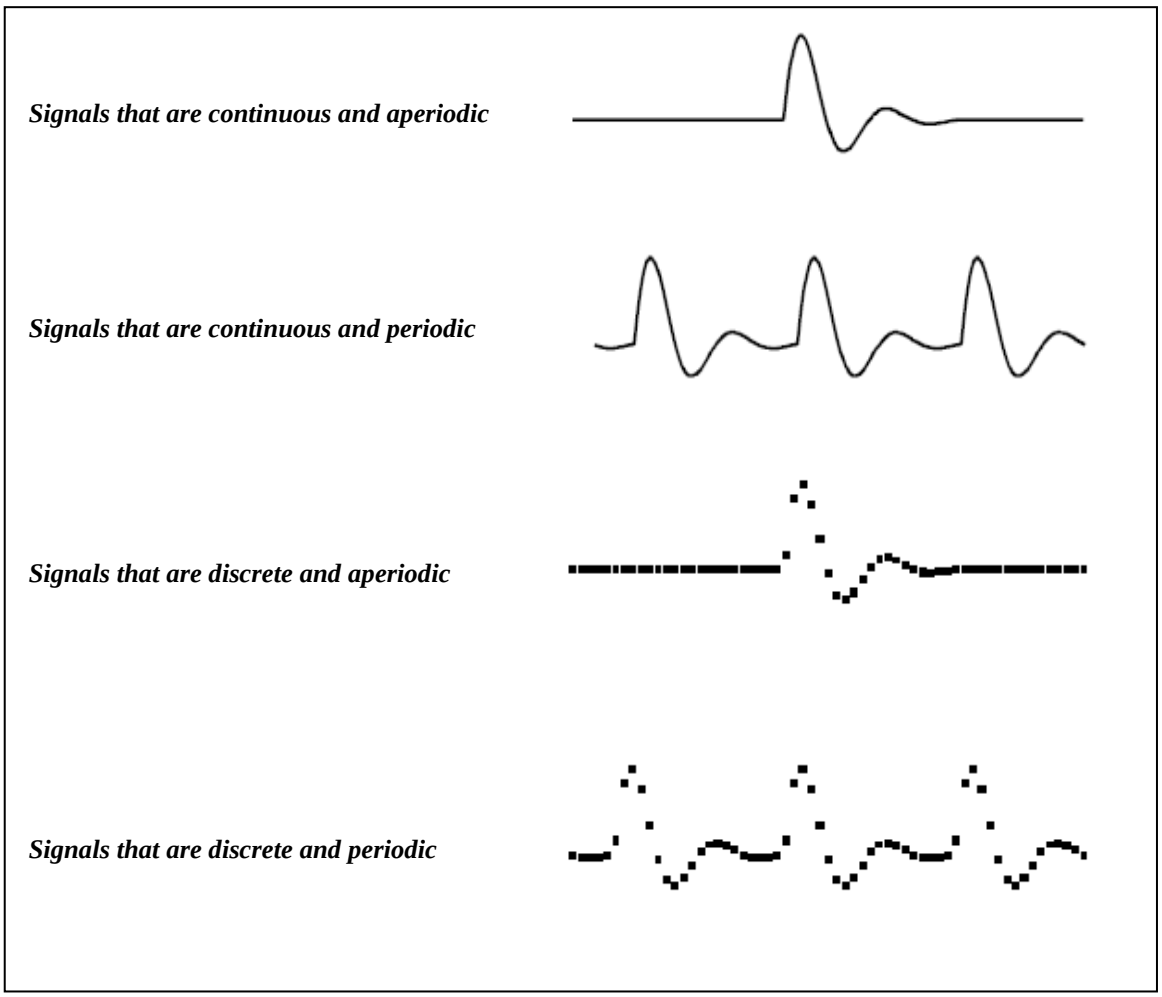


Fig (2.1) Signals types

In this case, the real part is *(cos x)* while the imaginary part is *(sin x)*, from this form we could rewrite equation (2.1) as [MSY99]:

$$F(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)[\cos(2\pi ux/N) - j\sin(2\pi ux/N)] \qquad (2.3)$$

For u=0, 1, 2, ..., N-1

The value of *F(u)* is complex number, with real part corresponding to the cosine term and the imaginary part corresponding the sine term, therefore it could be represent *F(u)* as $F_r(u) + j\,F_i(u)$, and each part could be computed separately, as follows [MSY99]:

$$F_r(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)\cos(2\pi ux/N) \quad (2.4)$$

$$F_i(u) = \frac{-1}{N}\sum_{x=0}^{N-1} f(x)\sin(2\pi ux/N) \quad (2.5)$$

For u=0, 1, 2, ..., N-1

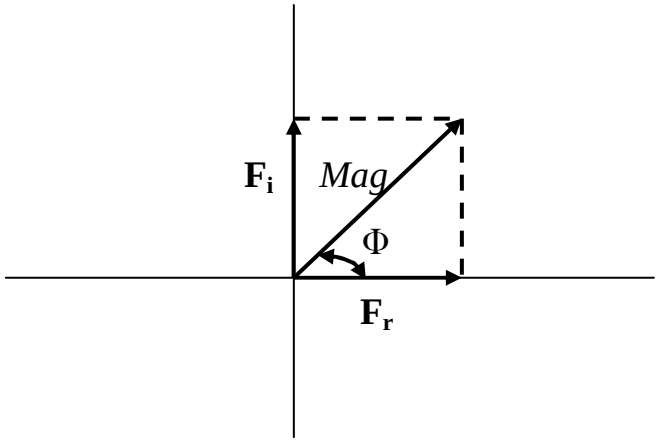
Where $F_r(u)$ is the real part, while $F_i(u)$ is the imaginary part. The magnitude and phase of the complex number $F(u)$ are determined according to the following [MSY99]:

$$Mag(u) = |F(u)| = \sqrt{F_r^{\,2}(u) + F_i^{\,2}(u)} \quad (2.6)$$

And

$$Phs(u) = \Phi(u) = \tan^{-1}\left(\frac{F_i(u)}{F_r(u)}\right) \quad (2.7)$$

The *Mag* is *magnitude* defined as the length of the vector starting from the origin and ending at the point on the complex plane, while the *Phs* is *phase* represented the angle between this vector and the positive x-axis [Smi99]. Could represent the coordinate for phase and magnitude as:



In addition, the equations to reconstruct the real and imaginary part from phase and magnitude are [MSY99]:

$$F_r(u) = Mag(u)Cos(Phs(u)) \quad (2.8)$$

And

$$F_i(u) = Mag(u)Sin(Phs(u)) \quad (2.9)$$

The transform from frequency domain to time domain is called *the Inverse Discrete Fourier Transform* given by [MSY99]:

$$F^{-1}(F(u)) = f(x) = \frac{1}{N}\sum_{u=0}^{N-1} F(u) e^{j2\pi ux/N} \qquad (2.10)$$

For x=0, 1, 2, ..., N-1

In many applications, the output data for DFT is complex and satisfy the symmetry in its magnitude and phase, this property allows for concentrating the determinations and plot tasks on just half of the period (i.e., half number of Fourier coefficients). This property is called *conjugate symmetry* [MSY99]*:*

$$F(N-u) = F(u), \qquad (2.11)$$

For u=1,....., N-1

Where, N is the number of samples.

The conjugate symmetry property implies that the real part and magnitude are even functions while the phase and imaginary part are odd functions. This property was used to simplify the analysis and to reduce the number of computations. Equation (2.11) could be expressed by the following alternative expressions [MSY99]:

$$F_r(N-u) = F_r(u) \qquad (2.12.a)$$

$$F_i(N-u) = -F_i(u) \qquad (2.12.b)$$

$$Phase(F(N-u)) = 360 - Phase(F(u)) \qquad (2.12.c)$$

$$Magnitude(F(N-u)) = Magnitude(F(u)) \qquad (2.12.d)$$

For u=1,....., N-1

2.8 Quantization Process

Quantization refers to the process of approximating the continuous set of values in the signal with a finite set of values to discrete values [Sal00]. Widrow's Quantization Theorem describes quantization theorem, by modeled, see figure (2.2), as the addition of a uniform distributed random signal ***e*** and the original unquantized signal ***q*** [Zol98].

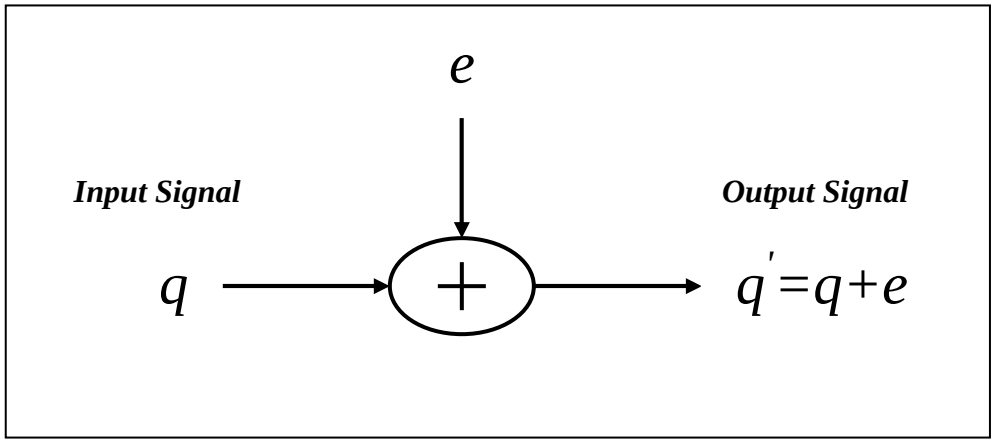


Fig (2.2) A quantization model

The quantizer is a function whose set of output values are discrete, and usually finite. Obviously, quantization is a process of approximation, and a good quantizer is one which represents the original signal with minimum loss or distortion. So can used the quantization error (q - q') as a measure of the optimality of the quantization process [Sal00].

Quantization type that deal with taking a single value and reducing the number of bits used to represent that value called *scalar quantization* and is most easily achieved by rounding. While *vector quantization* treats the entire vector value as a single entity and quantizes it by reducing the total number of bits required to represent the vector.

In addition, there are two ways to do quantization theorem: *uniform quantization* or *non-uniform quantization*. In uniform quantization, all the subdivisions into which the range is divided of equal width. In non-

uniform quantization, these quantization bins are not all of equal width [Umb98].

2.9 The Sound [Smi99]

Sound in natural is a series of air pressure fluctuations represented as analogue signal. The analogue sounds are waves detected by human ears within the range (20 Hz to 20 kHz). These waves are continuous signal in both time and amplitude.

In addition to the sound signal, there are various kinds of continuous signal; light intensity that changes with distance; voltage that varies over time; a chemical reaction rate that depends on temperature, etc. Analog-to-Digital Conversion (ADC) and Digital-to-Analog Conversion (DAC) are the processes that allow digital computers to interact with these signals everyday.

Sound in the computer is a graph of the change in air pressure over time and represent as digital sound. Digital sounds are discrete signal in time and amplitude. The unit to deal with digital sound is called *sample*. Digital signal is different from its continuous signal in two important aspects: *sampling*, and *quantization*. Both of these aspects restrict how much information a digital signal can contain.

2.9.1 Sampling

Sampling is the first step towards conversion signals from analog (continuous-time) to digital (discrete-time) domains. It consists of measuring the amplitude of the analog audio waveform at periodic intervals of time; figure (2.3) illustrates the sampling of analog signal. The time axis is divided by sampling process. The time between samples is called the sampling interval. The main concern is to

represent the original analog values with adequate precision. The measurement accuracy depends on the frequency at which the audio

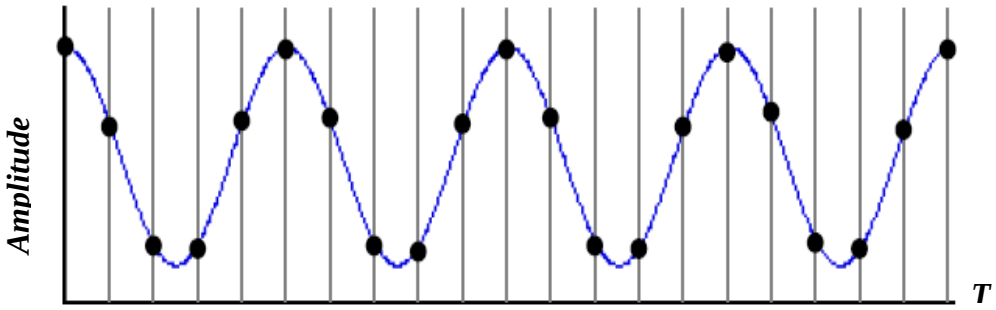


Fig (2.3) Sampling analog signal

signal is measured, or sampled. Nyquist and Shannon theorem states that the sampling frequency must be at least twice (preferably more than twice) the highest audio frequency being sampled [Rob01].

The precision at which the digitized signal will represent the continuous signal depends on two parameters of the digitization process. First, the rate at which that amplitude measurements are made (i.e., the *sampling rate* or *sampling frequency*). The unit of sampling rate is Hertz, this word refers to both sampling rate where it refers to samples per second, while for frequency the word Hertz refers to cycles per seconds. Second the number of bits used to represent each amplitude measurement (the *sample size*). Different models have been provided to represent the audio *samples* 8-bit per sample can resolve 256 ($=2^8$) different amplitude values; or 16-bit per sample can resolve 65,536 ($=2^{16}$) values.

When a sample is taken, the actual value is rounded to the nearest value that can be represented by the number of bits in a sample. Since the actual analog value of the signal amplitude at the time of sampling is usually not exactly equal to one of the discrete values that can be represented exactly by a sample, there is some error inherent in the process of digitization [Smi99].

Standard formats to store samples, make the way easier for software developers and equipment manufacturers to produce products that are less costly and more compatible with each other. Pulse Code Modulation (PCM) is a common method of storing and transmitting uncompressed digital audio. PCM is a straight representation of the binary digits (1s and 0s) of sample values. When PCM audio is transmitted, each "1" is represented by a positive voltage pulse and each "0" is represented by the absence of a pulse. Figure (2.4) shows a PCM signal format. PCM is still the most widely used digital representation for representation of audio signal [Tim98].

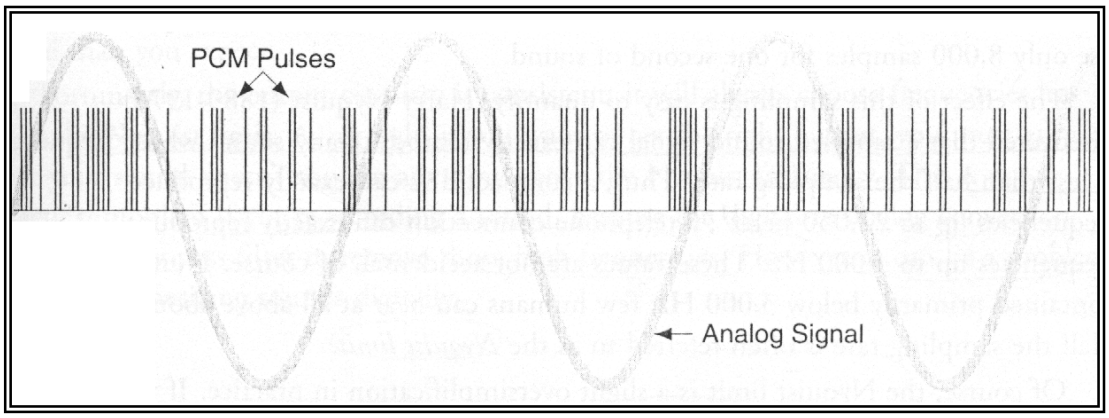


Fig (2.4) Pulse code modulation

2.9.2 Quantization

The term *quantization* refers to convert the amplitude axis from continuous to discrete values. The two variables are the two axes of an analog sound wave. The X axis is time, and the Y axis is voltage. Both of these axes must be divided into discrete values in order to store the analog signal in digital form. The time axis is divided by a sampling process. While the amplitude axis divided by quantization [Smi99], see figure (2.5).

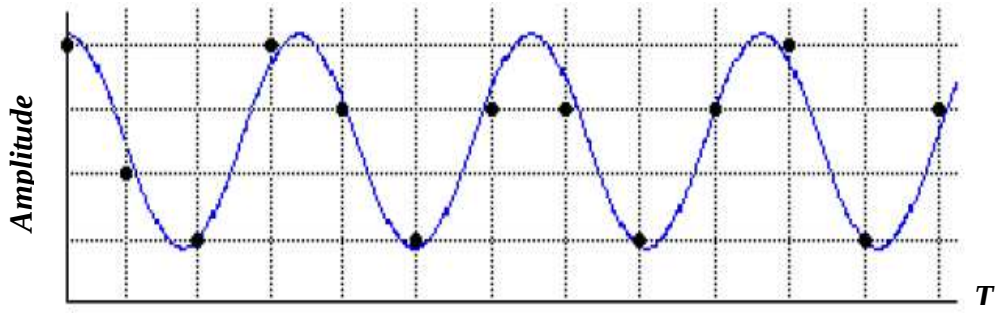


Fig (2.5) Signal quantization

The number of available values is determined by the number of bits (0's and 1's) used for each sample; also it is called **bit depth** or **bit resolution**. Each additional bit doubles the number of quantization levels (1-bit samples have two discrete values; 2-bit samples have four discrete values, etc.). When a sample is quantized, the instantaneous snapshot of its analog amplitude has to be rounded off to the nearest available digital value. This rounding-off process, called **approximation**. The digital signal is defined only at the point at which it is sampled, which is sometimes higher and sometimes lower than the original signal within the sampling interval. The smaller number of bits used per sample, greater distances the analog values need to be rounded off too. The difference between analog values and digital values is called *quantizing error* as shown in the figure (2.6) [Has01].

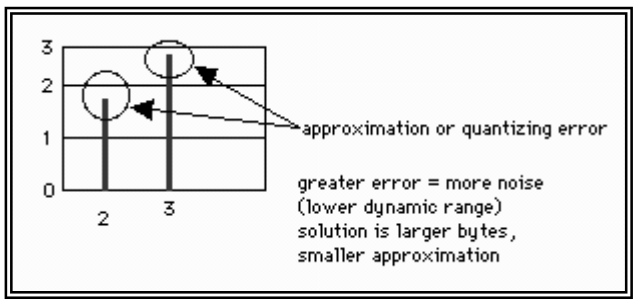


Fig (2.6) Quantization error

An increase in the magnitude of approximation errors cause an increase in the level of produced quantization noise. To reduce the ***digital***

noise we use larger sample sizes (greater bit depth), which in turn correspond to the dynamic range of the digital system, as shown in figure (2.7).

The use of high sampling frequency and more bits in quantization will produce better quality digital audio, but for the same length of audio, the file size will be much larger than the low quality audio [Has01].

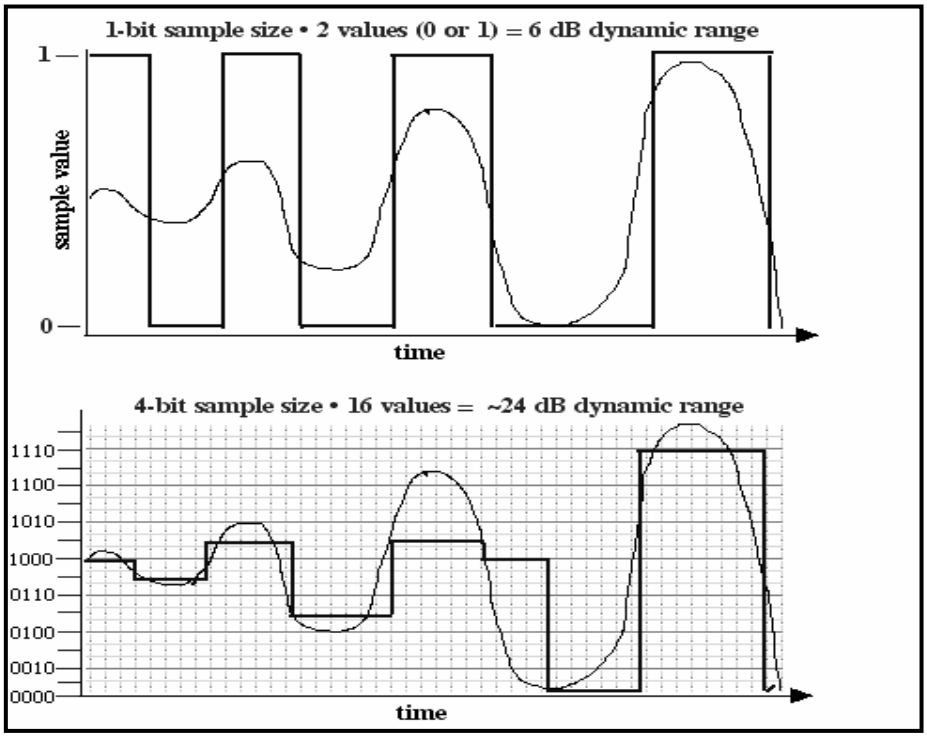


Fig (2.7) Quantization using one bit and two bits sampling resolution

2.10 Digital Audio Structure

Digital audio file has two main parts: the header and the audio data. The header is used to store information about the file, including the resolution, sampling rate and type of compression, etc. The header structure based on *chunk*s and *sub-chunks*. Each *Chunk* has a type, represented by a number of character called ChunkID, comes first in the *chunk*, followed by ChunkSize (long integer) is the size of the rest chunk

data (except the size of ChunkID, and ChunkSize), then comes the content of the chunk [Bor01].

Multimedia applications require the storage and management of a wide variety of data, including bitmaps, audio data, video data, and peripheral device control information. RIFF stands for *Resource Interchange File Format* provides a way to store all these varied types of data. RIFF file may contain different types of data, these types indicated by the file extension. Examples of data that may be stored in RIFF files are [Web03]:

1. Audio/visual interleaved data (.AVI).
2. Waveform data (.WAV).
3. Bitmapped data (.RDI).
4. MIDI information (.RMI).
5. Color palette (.PAL).
6. Multimedia movie (.RMN).
7. Animated cursor (.ANI).
8. A bundle of other RIFF files (.BND).

The entire RIFF file is a big chunk; it may contain various types of sub-chunks. The first thing in the contents of the RIFF chunk is the form type which describes the overall type of the file's contents such as WAV, AVI, etc [Web03].

Windows Audio Visual (WAV) audio files are one of the common formats used to store and play digital audio data. WAV file is usually coded in PCM format, which means uncompressed file; it takes up a lot of space. Sometimes, wave files may be coded in other formats, including MP3 format, which mean compressed file. WAV file supports various characteristics; sampling resolution, number of channels, and

compression type, etc. There are two *sub-chunks* that are required in order to successfully save sampled audio waveforms [Bor01]:

1. The *fmt chunk*, specifying the data format.
2. The *data chunk*, containing the actual sample data.

The main advantage of using this chunk structure is that when parsing a WAVE file we don't need to interpret every chunk type but we could skip over the sub-chunks we don't need by reading the ChunkID and ChunkSize parameter, after identifying the type of the sub-chunk (using ChunkID) and found it is not the wanted one, we can skip this chunk by using the second parameter.

The structure of wave audio file consists at least of two common chunks. These two sub-chunks are fmt and data chunks, their contents are described in tables (2.2), (2.3) [Web03].

The structure of fmt sub-chunk is described in table (2.2), the size of this sub-chunk usually equal to 16 bytes, but it was noticed that some sound recorder programs use 18 bytes instead 16 bytes (i.e., there are two additional bytes). The sub-ChunkID field consists of four characters (4 bytes), and the field of sub-chunk size is long integer (4 bytes) type, in case of the existence of the additional two bytes, they should by-passed.

Table (2.1) The RIFF Header Chunk

Offset	***Size (Byte)***	***Field name***	***Contents Description***
0	4	Chunk ID	Contains the letters "RIFF" in ASCII
4	4	Chunk Size	The size of WAV file in bytes minus 8 bytes (4 bytes to Chunk ID and 4 bytes to Chunk Size)
8	4	Format	Contains the letters "WAVE"

Table (2.2) The fmt Sub-Chunk

Offset	***Size (Byte)***	***Field name***	***Contents Description***
12	4	Sub-Chunk1 ID	Contains the letters *"fmt "*
16	4	Sub-Chunk1 Size	This number describe the of the rest sub-chunk data: audio format (2bytes) + num. channels (2bytes) + sample rate (4bytes) + bytes/sec (4bytes) + block alignment (2bytes) + bits/sample (2bytes). This is usually 16.
20	2	Audio Format	Type of WAVE format. PCM=1 value other than 1 indicate some form of compression
22	2	Channel Numbers	Channels: Mono=1, Stereo=2, etc.
24	4	Sample Rate	Samples per second. e.g., 8000,22000, 44100, etc.
28	4	Byte Rate	Bytes per Second = Sample Rate × Channel Numbers × BitsPerSample/8
32	2	Block Align	The number of bytes for one sample including all channels. Or = Channel Numbers × BitsPerSample/8
34	2	Bits Per Sample	Value 8 = 8 bits, Value 16 = 16 bits, etc.

Table (2.3) The Data Sub-Chunk

Offset	***Size (Bytes)***	***Field name***	***Contents***
36	4	Sub-Chunk2 ID	Contains the letters "data"
40	4	Sub-Chunk2 Size	Size in bytes refers to the rest sub-chunk, mean the actual data sound.
44	*	Data	The sound data.

Chapter Three

The Proposed System Implementation

3.1 The System Model

The general structure of the proposed system is illustrated in figure (3.1). It consists of two basic modules: hiding and extraction modules. The input to this system are the cover file (wave file), and secret file (binary file). These inputs are processed in the hiding part with various operations to produced stego wave file. The stego audio entered to extraction stage is processed through a set of operations to retrieve the secret data.

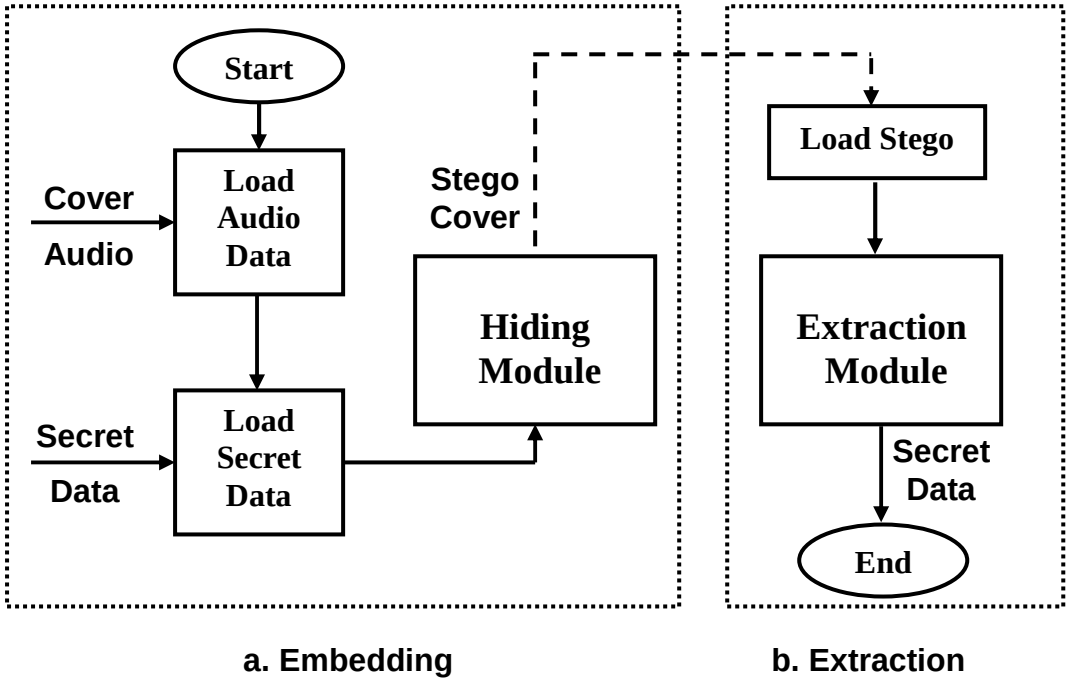


Fig (3.1) The general system model

Each module (i.e. hiding and extraction) consists of several processes, working systematically to lead to the final result. However, some of these processes are replicated in hiding and extraction modules,

and these processes are load wave file, Fourier transform, inverse Fourier transform and quantization process.

3.2 Load the Cover File

The first step in the embedding module involved with reading both cover and secret files. In this research, the type of the used cover file was audio file of type waveform (.wav). The structure of this wave file, was explained in section (2.11), consists of a sequence of chunks, the chunk has three parts; ChunkID, CunkSize and data. When the audio wave file is opened as a first step, the header of the file must read, it contains the main chunk *"RIFF chunk"*, and the sub-chunk *"fmt"*. Then, as a second step, the sub-chunk data must be loaded. For more details about the structure of these chunks, see tables (2.1), (2.2) and (2.3). The required properties about the audio samples are listed in "RIFF" chunk and *"fmt"* sub-chunk, while the actual data are given in "data" sub-chunk.

The wave data either stored as one channel, called *mono,* or stored in two channels, called *stereo*. The mechanism followed in this research implies handling the contents of each channel separately whatever is the data type.

The audio samples resolution (i.e., number of bits per sample) is either 8 bits (ranging from 0 to 255), or 16 bits (ranging from -32768 to 32768). This may cause confusion when dealing with both kinds of sample resolutions, because to handle both cases two structures for system must be established, the first one to handle the 8-bits samples, and the second to handle the 16-bits samples. To avoid this confusion, and to unify the handles of both cases the range of the sample values were mapped to be between 0 and 255. Therefore, the value of each sample was represented by 8-bits (whatever its original sample resolution is 8 or 16 bits), in this case, an array of byte type was originated to represent the

audio cover data. Pseudo code list (3.1) shows the implemented steps to read and map the cover wave file.

When the original audio cover data has sample resolution 16-bits and it is mapped to 8-bits resolution the quality of the audio signal will slightly degraded, and this may be considered as signal degradation. But, in case of steganography applications, this decrease in sample resolution will not be consider as a deviancy because the quality of the cover data will somehow reduced due to data-embedding process. The level of distortion cause by data embedding stage is, mostly, higher than the subjective degradation cases by reducing the samples resolution from 16 to 8-bits.

Code List (3.1): Load the cover wave file

Input:

CovName: The name of secret file

Output:

Hdr: Header of cover file

DataSize: The size of the real cover data

Cov: Array of bytes contains the data sample of the audio file

Begin

Open the cover file "CovtName"

Read Hdr from file // The header of wave file

Extract ChannelNo from Hdr

Extract SampleResolution from Hdr // Bits Per Sample

Set L ← 37 //jump to location after header types until data Chunck

Do

Seek to L in file

Read ChunkID

Read DataSize

Set L ← L + DataSize

While ChunkID <> "data"

If ChannelsNo = 1 then

Continue

```
If SamplResolution = 8 then
    Get Vectr1 from File    //*Vectr1 is an array of bytes represent
                                audio data *//
    For each sample i in Vectr1
      Set  Cov(i) ← Vectr1(i)
    End loop
ElseIf SamplResolution = 16 then
    Set DataSize ← DataSize \ 2
    Set  R ← 128 / 32767
    Get Vectr2 from file            //*Vectr2 is an array of integer
                                represent audio data*//
    For each sample i in Vectr2
      Set Cov(i) ← (Vectr2(i) × R) + 128  //*convert to range
                                [0..255]*//
    End loop
  EndIf
ElseIf  ChannelNo = 2 then
   If SamplResolution = 8 then
      Set  DataSize ← DataSize \ 2
      Get Vectr1 from File
      For each sample i in Vectr1
        Set  Cov(i) ← Vectr1(2 × i)
      End loop
   ElseIf SamplResolution = 16 then
      Set  DataSize ← DataSize \ 4
      Set  R ← 128 / 32767
      Get Vectr2 from file
      For each sample i in Vectr2
        Set  Cov(i) ← (Vectr2(2 × i) × R) + 128
      End loop
   EndIf
EndIf
End.
```

3.3 Load the Secret Data

The secret file in this system is treated as a binary file, whatever is its type (like, image, audio, text…). Therefore, after reading the secret data as an array of byte type (i.e., the values ranging between 0 and 255), the sequence of bytes must converted to binary sequence, to get the final data as a sequence of ones and zeros. The code list (3.2) shows the implemented steps to load the secret data while the code list (3.3) illustrates the process of the binary conversion.

Code List (3.2) Load the secret file

```
Input:
        ScrtName: Secret file Name
Output:
        OrgScrtLen: Size of secret file
        OrgScrt: Secret data as an array of bytes
Begin
        Open the secret file "ScrtName"
        Get OrgScrtLen                  //* The length of secret file
        Get secret data OrgScrt         //* OrgScrt[0..OrgScrtLen] as
                                            an  array of Bytes.
End.
```

Code List (3.3): Binary conversion

Input:

OrgScrt: Secret data as an array byte

OrgScrtLen: Size of secret file

Output:

Scrt: Binary secret data as an array byte

ScrtLen: New size of secret data

Begin

Set K ← 0

For each byte value i in OrgScrt

For each bit j in byte // from 0 to 7

Set Scrt(K+j) ← OrgSecrt(i) AND (2^ j)

Set K ← K + 8

End loop

Set ScrtLen ← OrgScrtLen × 8

End.

3.4 The Hiding Module

The hiding part consists of a sequence of processes on the cover file. As a first stage, the cover audio data are partitioned into blocks of samples. The block size must previously be predefine by the user, and must be not so large to avoid the occurrence of high computational complexity. In this research, the block size was chosen 8, 12, 16, 24, or 32. In the remaining part of this chapter the block size is denoting by N. Beside to partitioning the audio data, the following processes were applied:

1. Voiced and unvoiced blocks classification.
2. Quick Fourier transform (QFT) and inverse quick Fourier transform (IDFT).
3. Uniform quantization.
4. Hiding in Selected Blocks.

3.4.1 The Voiced / Unvoiced Sound Classification

Sound consists of acoustic pressure waves, for example the speech is created by the voluntary movements of anatomical structures in the human speech production system. As the diaphragm forces air through the system, these structures are able to generate and shape a wide variety of waveforms. These waveforms can be broadly categorized into voiced and unvoiced sound.

The adopted embedding mechanism in one of proposed hiding methods in this research, depends on hiding in the voice area of the cover sound and ignore unvoiced area. Therefore, the voiced regions in the cover sound must isolate from unvoiced area. The implemented and unvoiced classification was based on measuring the voice power for each block of the cover wave data, and compare this power with a predefined threshold (*Thr*) to decide if the tested block is considered voiced block or not. In addition, if the determined power of the voiced block is too close to the threshold value, then the power of such block is increased, while it is decreased for the case of unvoiced block. This step is performed to avoid the possibility of power shifting due to the further processing imposed on the voiced blocks (like, secret data embedding, and the rounding process that follows inverse Fourier transform). This shift may cause a change in status of the auto-classified block from voiced to unvoiced. The adopted kind of power shifting was done by checking the difference between the power of voiced block and the threshold (*Thr*) value. If this difference is less than an alpha value (α), then the tested block is send to power shifting process. If the result of comparison is less than (α) then multiply all the samples of the block by a predefined parameter (β). The value of (β) should be grater than 1 for voiced blocks, and it is less than 1 for unvoiced blocks. Pseudo code (3.4) illustrates the steps of checking the voiced and unvoiced blocks.

Code List (3.4): Voice/Unvoiced block checking

Input:

N: The block Size. *// N = 8, 12, 16, 24, 32*

WavBlk: N samples of wave audio.

Thr: Predefined parameter determined the bounded power voiced and unvoiced Block.

α: This parameter determined the bounded of power shifting area.

β: This parameter determined the power shifting area.

Output:

Flag: Boolean value

Begin

Set T ← Thr × Thr × N

Set Pw ← 0

For each sample i in WavBlk

Set Pw ← Pw + (WavBlk(i)-128) ^ 2

End loop

If Abs(Pw - T) < α Then

If Pw >= T Then

For each sample i in WavBlk

Set WavBlk(i) ← β × (WavBlk(i) – 128)+128

End loop

Set Flag ← True

Else

For each sample i in WavBlk

Set WavBlk(i) ← β × (WavBlk(i) – 128)+128

End loop

Set Flag ← False

End If

Else

If Pw >= T then Set Flag ← True else Set Flag ← False.

End If

End.

3.4.2 The Fourier Transform Implementation

The basic step in the proposed system to hide secret data in audio is Fourier transform. Here the input data to the Fourier transform are the samples of the cover voiced block, these samples are considered as the waveform representation in *time domain* because the samples taken over time. The output of Fourier transform is considered as the representation in *frequency domain*. The term *frequency domain* is used to describe the signal in terms of amplitudes and phases of a series of fundamental sinusoidal and cosinusoidal waves. The *frequency domain* contains exactly the same information as the *time domain*; it is just in a different form. If you know one domain, you can calculate the other. The process of calculating the frequency domain from time domain is called the *forward DFT*, or simply, the *DFT*. In addition, the calculation of the time domain from frequency domain called the *inverse DFT*, equations (2.1) and (2.8) described these two mapping forms.

The number (N) of samples of each voiced block passed through Fourier transform represent the size of the block, and it may not be of power (2^n). The values of *N* taken in this work are 8, 12, 16, 24, and 32. The main reason behind choosing *N* as a small number is to make the computation time as small as possible, and to avoid the occurrence of a drastic changes in the characteristics of the audio signal that hold in the voiced blocks.

For a signal consists of N-samples the calculation of DFT coefficients requires the number 2N(N-1) of multiplications and additions operations. From equation (2.1) we can notice that the determination of each complex coefficient, F(u), require N multiplication and N-1 addition operations. So, the determination of N-complex coefficients require 2N(N-1) multiplication and addition, taken into consideration that the determination of DC-coefficient doesn't need multiplication operations.

When the value of N is increased the number of operations is increased and the computation time will become longer.

In this research, the equation of Fourier transform was expanded for certain values of N (i.e., 8, 12, 16, 24, 32), then the expanded equations were simplified by taking the benefits of the existing symmetry in the coefficients, and by converting the float operations (addition and multiplications) to integer operation. This strategy is called *Quick Fourier Transform (QFT).*

In chapter two, it is mentioned that involved equations of Fourier transform could be written as two set of equations; the first set is to determine the real part coefficients (F_r), see equation (2.4), and the second set is to determine the imaginary part coefficient (F_i), see equation (2.5).

The first step in the quick Fourier transform (QFT) is the calculation of involved *cosine* and *sine* functions. Because these functions are repeatedly computed with each term in Fourier transform equations, taking into consideration that the transform should applied over all voice blocks, which means that the values of these function are required too many times. Therefore, as a first step; two tables are constructed each one contains all possible needed values of the kernel functions (either cosine or sine).

The size of each kernel table is (N×N), because there are N coefficients of (F_r) and (F_i) in each transform equation, and there are N transform equations, see equations (2.4) and (2.5).

Since, the range of values of cosine and sine functions is between [-1, 1]. So all the values listed in kernel tables are float and these values could be considered as constants in the calculation of the Fourier transform for certain N-value. Since the application of mathematical operations (addition, subtraction, multiplication, etc…) between floating

numbers implies more complexity in comparison with complexity of same operations applied on integer numbers. In addition, this high complexity will make the overall time required to perform the overall involved computation of Fourier transform is a long time. To speed up the calculations all involved operations were mapped to be of integer type, this step is done by mapping all values of cosine and sine function from floating number to integer number. The converted values are stored as signed long integers whose values extend from (-2147483648) to (2147483647). The conversion from float to integer number was easily done by applying the following linear mapping (or approximation) process:

$$I = round\ (fn \times 2^{10}) \qquad (3.1)$$

Where: *f is the floating number.*

I is the corresponding integer number.

Although the above mapping involves an approximation operation (i.e., round operation), it does not cause significant effects on the accuracy of the determined Fourier coefficients (the order of the error caused by this approximation will be around 0.001). Due to this integer mapping, the final values of Fourier coefficients are multiplied by 2^{-10}. The code list (3.5) illustrates the implemented steps to determine the kernel function values.

To illustrate the reduction performed on Fourier transform computations by using a set of expanded equations with low computational complexity, let us consider the case when the transformed audio block size consist of eight samples (i.e., N = 8). Later the final reduced Fourier equations for the cases (N = 12, 16, 24, 32) will be given.

Code List (3.5): kernel table construction

Input:

N: The audio block size

Output:

C: tables of cosine data

S: table of sine data

Begin

Set Pi ← 4 × tan^{-1}(1)

Set A ← 2×Pi / N

For each index u in N

Set uu ← u × A

For each index x in N

Set B ← uu × x

C(u,x) = Cos(B) × 1024

S(u,x) = Sin(B) × 1024

End loop

End loop

End.

The steps followed to reduce to transform equations when the block size is eight are the followings:

Step1: Construct the tables of the values of kernel functions, (i.e. cosine *C(i,j)* and sine *S(i,j)*). The determined values that listed in cosine table *C(i,j)* are:

$$C(i,j)=\begin{bmatrix} 1024 & 1024 & 1024 & 1024 & 1024 & 1024 & 1024 & 1024 \\ 1024 & 724 & 0 & -724 & -1024 & -724 & 0 & 724 \\ 1024 & 0 & -1024 & 0 & 1024 & 0 & -1024 & 0 \\ 1024 & -724 & 0 & 724 & -1024 & 724 & 0 & -724 \\ 1024 & -1024 & 1024 & -1024 & 1024 & -1024 & 1024 & -1024 \\ 1024 & -724 & 0 & 724 & -1024 & 724 & 0 & -724 \\ 1024 & 0 & -1024 & 0 & 1024 & 0 & -1024 & 0 \\ 1024 & 724 & 0 & -724 & -1024 & -724 & 0 & 724 \end{bmatrix}$$

While, the determined values that listed in sine table *S(i,j)* are:

$$S(i,j)=\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 724 & 1024 & 724 & 0 & -724 & -1024 & -724 \\ 0 & 1024 & 0 & -1024 & 0 & 1024 & 0 & -1024 \\ 0 & 724 & -1024 & 724 & 0 & -724 & 1024 & -724 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -724 & 1024 & -724 & 0 & 724 & -1024 & 724 \\ 0 & -1024 & 0 & 1024 & 0 & -1024 & 0 & 1024 \\ 0 & -724 & -1024 & -724 & 0 & 724 & 1024 & 724 \end{bmatrix}$$

Step2: Expand equations (2.4) and (2.5), to get the eight equations of the real (F_r) and imaginary (F_i) coefficients:

$$\begin{bmatrix} F_r(0) \\ F_r(1) \\ F_r(2) \\ F_r(3) \\ F_r(4) \\ F_r(5) \\ F_r(6) \\ F_r(7) \end{bmatrix} = \begin{bmatrix} 1024f_0 & 1024f_1 & 1024f_2 & 1024f_3 & 1024f_4 & 1024f_5 & 1024_{f6} & 1024f_7 \\ 1024f_0 & 724f_1 & 0f_2 & -724f_3 & -1024f_4 & -724f_5 & 0f_6 & 724f_7 \\ 1024f_0 & 0f_1 & -1024f_2 & 0f_3 & 1024f_4 & 0f_5 & -1024f_6 & 0f_7 \\ 1024f_0 & -724f_1 & 0f_2 & 724f_3 & -1024f_4 & 724f_5 & 0f_6 & -724f_7 \\ 1024f_0 & -1024f_1 & 1024f_2 & -1024f_3 & 1024f_4 & -1024f_5 & 1024f_6 & -1024f_7 \\ 1024f_0 & -724f_1 & 0f_2 & 724f_3 & -1024f_4 & 724f_5 & 0f_6 & -724f_7 \\ 1024f_0 & 0f_1 & -1024f_2 & 0f_3 & 1024f_4 & 0f_5 & -1024f_6 & 0f_7 \\ 1024f_0 & 724f_1 & 0f_2 & -724f_3 & -1024f_4 & -724f_5 & 0f_6 & 724f_7 \end{bmatrix}$$

$$\begin{bmatrix} Fi(0) \\ Fi(1) \\ F_i(2) \\ F_i(3) \\ F_i(4) \\ F_i(5) \\ F_i(6) \\ F_i(7) \end{bmatrix} = \begin{bmatrix} 0f_0 & 0f_1 & 0f_2 & 0f_3 & 0f_4 & 0f_5 & 0_{f6} & 0f_7 \\ 0f_0 & 724f_1 & 1024f_2 & 724f_3 & 0f_4 & -724f_5 & -1024f_6 & -724f_7 \\ 0f_0 & 1024f_1 & 0f_2 & -1024f_3 & 0f_4 & 1024f_5 & 0f_6 & -1024f_7 \\ 0f_0 & 724f_1 & -1024f_2 & 724f_3 & 0f_4 & -724f_5 & 1024f_6 & -724f_7 \\ 0f_0 & 0f_1 & 0f_2 & 0f_3 & 0f_4 & 0f_5 & 0f_6 & 0f_7 \\ 0f_0 & -724f_1 & 1024f_2 & -724f_3 & 0f_4 & 724f_5 & -1024f_6 & 724f_7 \\ 0f_0 & -1024f_1 & 0f_2 & 1024f_3 & 0f_4 & -1024f_5 & 0f_6 & 1024f_7 \\ 0f_0 & -724f_1 & -1024f_2 & -724f_3 & 0f_4 & 724f_5 & 1024f_6 & 724f_7 \end{bmatrix}$$

The symbol *f* refers to samples values in time domain (or audio data block).

From the above two sets of equations, there is a symmetry in the values of coefficients belong to the same row, and coefficients belong to different blocks. In chapter two, the meaning of the symmetry features

was given and it is represented by certain equations. From equations (2.12.a) and (2.12.b) the following symmetry characteristic is deduced:

$F_r(8-u) = F_r(u)$

$F_i(8-u) = -F_i(u)$

Where, u = 1, 2, 3, … , 7

Therefore, there is need to compute just the coefficients $F_r(u)$ and $F_i(u)$ for $(0 \leq u \leq N/2)$.

From the above two sets of equations, it is obvious that there are two kind of intra equation symmetry. The first kind occurs within the equations of the even coefficients, and the second kind occurs within the coefficients of the odd Fourier coefficients. Each kind of these two symmetries was handled in a proper way to simplify the above sets of equations, and finally to get the sets of quick Fourier transform equations. According to this method, most of the involved loops are ignored.

In the following paragraph the final form of the simplified (quick) Fourier transform for the cases (N = 8, 12, 16, 24, 32) are shown in tables (3.1 to 3.5) respectively. To less number of operations, the constant *W* is used of the term (1 / (N × 1024)) in the expanded equations, and *V* used of the term (1 / N).

Table (3.1) Quick Fourier Transform (N = 8)

Real Coefficients	Intermediate	$\mathbf{A_{04p} = f_0 + f_4}$, $\mathbf{A_{04m} = f_0 - f_4}$, $\mathbf{A_{17} = f_1 + f_7}$, $\mathbf{A_{26} = f_2 + f_6}$, $\mathbf{A_{35} = f_3 + f_5}$ $\mathbf{R_1 = 1024 \times A_{04m}}$, $\mathbf{R_2 = A_{04p} + A_{26}}$, $\mathbf{R_3 = A_{17} + A_{35}}$, $\mathbf{R_4 = 724 \times (A_{17} - A_{35})}$ $\mathbf{R_5 = A_{04p} - A_{26}}$
	Final	$\mathbf{F_r(0) = (R_2 + R_3) \times V}$, $\mathbf{F_r(1) = (R_1 + R_4) \times W}$ $\mathbf{F_r(2) = R_5 \times V}$, $\mathbf{F_r(3) = (R_1 - R_4) \times W}$ $\mathbf{F_r(4) = (R_2 - R_3) \times V}$ $\mathbf{F_r(N - i) = F_r(i)}$ **For i = 1, 2, 3**

Imaginary Coefficients	Intermediate	$B_{17} = f_1 - f_3$, $B_{26} = f_2 - f_6$, $B_{35} = f_3 - f_5$ $G_1 = 724 \times (B_{17} + B_{35})$, $G_2 = B17 - B35$ $G_3 = 1024 \times B_{26}$
	Final	$F_i(0) = 0$, $F_i(1) = -(G1 + G3) \times W$ $F_i(2) = -G2 \times V$, $F_i(3) = -(G1 - G3) \times W$ $F_i(4) = 0$, $F_i(N - i) = -F_i(i)$ for i = 1, 2, 3

Table (3.2) Quick Fourier Transform (N = 12)

Real Coefficients	Intermediate	$A_{06p} = f_0 + f_6$, $A_{06m} = f_0 - f_6$, $A_{111} = f_1 + f_{11}$, $A_{210} = f_2 + f_{10}$, $A_{39} = f_3 + f_9$, $A_{48} = f_4 + f_8$, $A_{57} = f_5 + f_7$ $R = 1024 \times A_{06m}$ $R_{p0} = A_{06p} + A_{39}$, $R_{m0} = A_{06p} - A_{39}$ $R_{p1} = A_{210} + A_{48}$, $R_{m1} = A_{210} - A_{48}$ $R_{p2} = A_{111} + A_{57}$, $R_{m2} = 887 \times (A_{111} - A_{57})$ $T1 = R + 512 \times R_{m1}$, $T2 = R_{p2} - R_{p1}$, $T3 = R_{p1} + R_{p2}$
	Final	$F_r(0) = (R_{p0} + T3) \times V$, $F_r(1) = (T1 + R_{m2}) \times W$ $F_r(2) = (1024 \times R_{m0} + 512 \times T2) \times W$ $F_r(3) = (A_{06m} - R_{m1}) \times V$ $F_r(4) = (1024 \times R_{p0} - 512 \times T3) \times W$ $F_r(5) = (T1 - R_{m2}) \times W$, $F_r(6) = (R_{m0} - T2) \times V$ $F_r(N - i) = F_r(i)$ for I = 1, 2, … , 5
Imaginary Coefficients	Intermediate	$B_{111} = f_1 - f_{11}$, $B_{210} = f_2 - f_{10}$, $B_{39} = f_3 - f_9$, $B_{48} = f_4 - f_8$, $B_{57} = f_5 - f_7$ $G_{p1} = B_{111} + B_{57}$, $G_{m1} = B_{111} - B_{57}$ $G_{p2} = 887 \times (B_{210} + B_{48})$, $G_{m2} = B_{210} - B_{48}$ $T1 = 512 \times G_{p1} + 1024 \times B_{39}$
	Final	$F_i(0) = 0$, $F_i(1) = -(T1 + G_{p2}) \times W$ $F_i(2) = -(887 \times (G_{m1} + G_{m2})) \times W$ $F_i(3) = -(G_{p1} - B_{39}) \times V$ $F_i(4) = -(887 \times (G_{m1} - G_{m2})) \times W$ $F_i(5) = -(T1 - G_{p2}) \times V$, $F_i(6) = 0$ $F_i(N - i) = -F_i(i)$ for i = 1, 2, … , 5

Table (3.3) Quick Fourier Transform (N = 16)

Real Coefficients	Intermediate	$A_{08p} = f_0 + f_8$, $A_{08m} = f_0 - f_8$, $A_{115} = f_1 + f_{15}$, $A_{214} = f_2 + f_{14}$, $A_{313} = f_3 + f_{13}$, $A_{412} = f_4 + f_{12}$, $A_{511} = f_5 + f_{11}$, $A_{610} = f_6 + f_{10}$, $A_{79} = f_7 + f_9$ $R = 1024 \times A_{08m}$ $R_{p0} = A_{08p} + A_{412}$, $R_{m0} = A_{08p} - A_{412}$ $R_{p1} = A_{214} + A_{610}$, $R_{m1} = 724 \times (A_{214} - A_{610})$ $R_{p2} = A_{115} + A_{79}$, $R_{m2} = A_{115} - A_{79}$ $R_{p3} = A_{313} + A_{511}$, $R_{m3} = A_{313} - A_{511}$ $T1 = R_{p0} + R_{p1}$, $T2 = R_{p2} + R_{p3}$ $T3 = R + R_{m1}$, $T4 = 946 \times R_{m2} + 392 \times R_{m3}$ $T5 = 1024 \times R_{m0}$, $T6 = 724 \times (R_{p2} - R_{p3})$ $T7 = R - R_{m1}$, $T8 = 392 \times R_{m2} - 946 \times R_{m3}$ $T9 = R_{p0} - R_{p1}$
	Final	$F_r(0) = (T1 + T2) \times V$, $F_r(1) = (T3 + T4) \times W$ $F_r(2) = (T5 + T6) \times W$, $F_r(3) = (T7 + T8) \times W$ $F_r(4) = T9 \times V$, $F_r(5) = (T7 - T8) \times W$ $F_r(6) = (T5 - T6) \times W$, $F_r(7) = (T3 - T4) \times W$ $F_r(8) = (T1 - T2) \times V$ $F_r(N - i) = F_r(i)$ for $i = 1, 2, \dots, 7$
Imaginary Coefficients	Intermediate	$B_{115} = f_1 - f_{15}$, $B_{214} = f_2 - f_{14}$, $B_{313} = f_3 - f_{13}$, $B_{412} = f_4 - f_{12}$, $B_{511} = f_5 - f_{11}$, $B_{610} = f_6 - f_{10}$, $B_{79} = f_7 - f_9$ $G_{p1} = B_{115} + B_{79}$, $G_{m1} = B_{115} - B_{79}$ $G_{p2} = B_{313} + B_{511}$, $G_{m2} = B_{313} - B_{511}$ $G_{p3} = 724 \times (A_{214} + A_{610})$, $G_{m3} = 1024 \times (A_{214} - A_{610})$ $G_4 = 1024 \times B_{412}$ $T1 = 392 \times G_{p1} + 946 \times G_{p2}$, $T2 = G_{p3} + G_4$ $T3 = 724 \times (G_{m1} + G_{m2})$, $T4 = 946 \times G_{p1} - 392 \times G_{p2}$ $T5 = G_{p3} - G_4$, $T6 = G_{m1} - G_{m2}$

	Final	$F_i(0) = 0$, $F_i(1) = -(T1 + T2) \times W$ $F_i(2) = -(T3 + G_{m3}) \times W$, $F_i(3) = -(T4 + T5) \times W$ $F_i(4) = -T6 \times V$, $F_i(5) = -(T4 - T5) \times W$ $F_i(6) = -(T3 - G_{m3}) \times W$, $F_i(7) = -(T1 - T2) \times W$ $F_i(8) = 0$ $F_i(N - i) = -F_i(i)$ for $i = 1, 2, \dots, 7$

Table (3.4) Quick Fourier Transform (N = 24)

Real Coefficients	Intermediate	$A_{012p} = f_0 + f_{12}$, $A_{012m} = f_0 - f_{12}$, $A_{123} = f_1 + f_{23}$, $A_{222} = f_2 + f_{22}$, $A_{321} = f_3 + f_{21}$, $A_{420} = f_4 + f_{20}$, $A_{519} = f_5 + f_{19}$, $A_{618} = f_6 + f_{18}$, $A_{717} = f_7 + f_{17}$, $A_{816} = f_8 + f_{17}$, $A_{915} = f_9 + f_{15}$, $A_{1014} = f_{10} + f_{14}$, $A_{1113} = f_{11} + f_{13}$ $R = 1024 \times A_{012m}$ $R_{p0} = A_{012p} + A_{618}$, $R_{m0} = A_{012p} - A_{618}$ $R_{p1} = A_{222} + A_{1014}$, $R_{m1} = 887 \times (A_{222} - A_{1014})$ $R_{p2} = A_{420} + A_{816}$, $R_{m2} = A_{420} - A_{816}$ $R_{p3} = A_{123} + A_{1113}$, $R_{m3} = A_{123} - A_{1113}$ $R_{p4} = A_{321} + A_{915}$, $R_{m4} = 724 \times (A_{321} - A_{915})$ $R_{p5} = A_{519} + A_{717}$, $R_{m5} = A_{519} - A_{717}$ $R_6 = R_{p1} + R_{p2}$, $R_7 = R_{m1} + 512 \times R_{m2}$, $R_8 = R_{p3} + R_{p5}$, $T1 = R_{p0} + R_6$, $T2 = R_8 + R_{p4}$, $T3 = R + R_7$, $T4 = 989 \times R_{m3} + Rm4 + 265 \times R_{m5}$, $T5 = 1024 \times R_{m0} + 512 \times (R_{p1} - R_{p2})$ $T6 = 887 \times (R_{p3} - R_{p5})$, $T7 = 1024 \times (A_{012m} - R_{m2})$, $T8 = 724 \times (R_{m3} - R_{m5}) - R_{m4}$ $T9 = 1024 \times R_{p0} - 512 \times R_6$ $T10 = 512 \times R_9 - 1024 \times R_{p4}$ $T11 = R - R_7$, $T12 = 265 \times R_{m3} - R_{m4} + 989 \times R_{m5}$, $T13 = R_{m0} - R_6$

	Final	$F_r(0) = (T1 + T2) \times V$, $F_r(1) = (T3 + T4) \times W$ $F_r(2) = (T5 + T6) \times W$, $F_r(3) = (T7 + T8) \times W$ $F_r(4) = (T9 + T10) \times W$, $F_r(5) = (T11 + T12) \times W$ $F_r(6) = T13 \times V$, $F_r(7) = (T11 - T12) \times W$ $F_r(8) = (T9 - T10) \times W$, $F_r(9) = (T7 - T8) \times W$ $F_r(10) = (T5 - T6) \times W$, $F_r(11) = (T3 - T4) \times W$ $F_r(12) = (T1 - T2) \times V$, $F_r(N - i) = F_r(i)$ for i = 1, 2, … , 11
Imaginary Coefficients	Intermediate	$B_{123} = f_1 - f_{23}$, $B_{222} = f_2 - f_{22}$, $B_{321} = f_3 - f_{21}$, $B_{420} = f_4 - f_{20}$, $B_{519} = f_5 - f_{19}$, $B_{618} = f_6 - f_{18}$, $B_{717} = f_7 - f_{17}$, $B_{816} = f_8 - f_{17}$, $B_{915} = f_9 - f_{15}$, $B_{1014} = f_{10} - f_{14}$, $B_{1113} = f_{11} - f_{13}$ $G_{p1} = B_{123} + B_{1113}$, $G_{m1} = B_{123} - B_{1113}$ $G_{p2} = 724 \times (B_{321} + B_{915})$, $G_{m2} = B_{321} - B_{915}$ $G_{p3} = B_{519} + B_{717}$, $G_{m3} = B_{519} - B_{717}$ $G_{p4} = B_{222} + B_{1014}$, $G_{m4} = B_{222} - B_{1014}$ $G_{p5} = 887 \times (B_{420} + B_{816})$, $G_{m5} = B_{420} - B_{816}$ $G_6 = 1024 \times B_{618}$, $G_7 = 512 \times G_{p4}$ $G_8 = G_{m1} + G_{m3}$, $T0 = G_{p7} + G_6$ $T1 = 265 \times G_{p1} + G_{p2} + 989 \times G_{p3}$ $T2 = T0 + G_{p5}$, $T3 = 512 \times G_8 + 1024 \times G_{m2}$ $T4 = 887 \times (G_{m4} + G_{m5})$, $T5 = G_{p2} + 724 \times (G_{p1} - G_{p3})$ $T6 = 1024 \times (G_{p4} - B618)$, $T7 = 887 \times (G_{m1} - G_{m3})$, $T8 = 887 \times (G_{m4} - G_{m5})$ $T9 = 989 \times G_{p1} - G_{p2} + 265 \times G_{p3}$ $T10 = T0 - G_{p5}$, $T11 = G_8 - G_{m2}$
	Final	$F_i(0) = 0$, $F_i(1) = -(T1 + T2) \times W$ $F_i(2) = -(T3 + T4) \times W$, $F_i(3) = -(T5 + T6) \times W$ $F_i(4) = -(T7 + T8) \times W$, $F_i(5) = -(T9 + T10) \times W$ $F_i(6) = -T11 \times V$, $F_i(7) = -(T9 - T10) \times W$ $F_i(8) = -(T7 - T8) \times W$, $F_i(9) = -(T5 - T6) \times W$ $F_i(10) = -(T3 - T4) \times W$, $F_i(11) = -(T1 - T2) \times W$ $F_i(12) = 0$, $F_i(N - i) = -F_i(i)$ for i = 1, 2, … , 11

Table (3.5) Quick Fourier Transform (N = 32)

Real Coefficients	Intermediate	$A_{016p} = f_0 + f_{16}$, $A_{016m} = f_0 - f_{16}$, $A_{131} = f_1 + f_{31}$, $A_{230} = f_2 + f_{31}$, $A_{329} = f_3 + f_{29}$, $A_{428} = f_4 + f_{28}$, $A_{527} = f_5 + f_{27}$, $A_{626} = f_6 + f_{26}$, $A_{725} = f_7 + f_{25}$, $A_{824} = f_8 + f_{24}$, $A_{923} = f_9 + f_{23}$, $A_{1022} = f_{10} + f_{22}$, $A_{1121} = f_{11} + f_{21}$, $A_{1220} = f_{12} + f_{20}$, $A_{1319} = f_{13} + f_{19}$, $A_{1418} = f_{14} + f_{18}$, $A_{1517} = f_{15} + f_{17}$ $R = 1024 \times A_{016m}$ $R_{p0} = A_{016p} + A_{824}$, $R_{m0} = 1024 \times (A_{016p} - A_{824})$ $R_{p1} = A_{230} + A_{1418}$, $R_{m1} = A_{230} - A_{1418}$ $R_{p2} = A_{428} + A_{1220}$, $R_{m2} = 724 \times (A_{428} - A_{1220})$ $R_{p3} = A_{626} + A_{1022}$, $R_{m3} = A_{626} - A_{1022}$ $R_{p4} = A_{131} + A_{1517}$, $R_{m4} = A_{131} - A_{1517}$ $R_{p5} = A_{329} + A_{1319}$, $R_{m5} = A_{329} - A_{1319}$ $R_{p6} = A_{527} + A_{1121}$, $R_{m6} = A_{527} - A_{1121}$ $R_{p7} = A_{725} + A_{923}$, $R_{m7} = A_{725} - A_{923}$ $R_{p8} = 724 \times (R_{p1} - R_{p3})$, $R_{p9} = R_{p4} - R_{p7}$ $R_{p10} = R_{p5} - R_{p6}$, $T1 = R_{p0} + R_{p1} + R_{p2} + R_{p3}$ $T2 = R_{p4} + R_{p5} + R_{p6} + R_{p7}$ $T3 = R + 946 \times R_{m1} + Rm_2 + 392 \times R_{m3}$ $T4 = 1004 \times R_{m4} + 851 \times R_{m5} + 569 \times R_{m6} + 200 \times R_{m7}$ $T5 = R_{m0} + R_{p8}$, $T6 = 946 \times R_9 + 392 \times R_{10}$ $T7 = R + 392 \times R_{m1} - R_{m2} - 946 \times R_{m3}$ $T8 = 851 \times R_{m4} - 200 \times R_{m5} - 1004 \times R_{m6} - 569 \times R_{M7}$ $T9 = 1024 \times (R_{P0} - R_{P2})$ $T10 = 724 \times (R_{m4} - R_{m5} - R_{m6} + R_{p7})$ $T11 = R - 392 \times R_{m1} - R_{m2} + 946 \times R_{m3}$ $T12 = 569 \times R_{m4} - 1004 \times R_{m5} + 200 \times R_{m6} + 851 \times R_{m7}$ $T13 = R_{m0} + R_8$, $T14 = 392 \times R_9 - 946 \times R_{10}$ $T15 = R - 946 \times R_{m1} + R_{m2} - 392 \times R_{m3}$ $T16 = 200 \times R_{m4} - 569 \times R_{m5} + 851 \times R_{m6} - 1004 \times R_{m7}$ $T17 = R_{p0} - R_{p1} + R_{p2} - R_{p3}$

Real Coefficients	Final	$F_r(0) = (T1 + T2) \times V$, $F_r(1) = (T3 + T4) \times W$ $F_r(2) = (T5 + T6) \times W$, $F_r(3) = (T7 + T8) \times W$ $F_r(4) = (T9 + T10) \times W$, $F_r(5) = (T11 + T12) \times W$ $F_r(6) = (T13 + T14) \times W$, $F_r(7) = (T15 + T16) \times W$ $F_r(8) = T17 \times V$, $F_r(9) = (T15 - T16) \times W$ $F_r(10) = (T13 - T14) \times W$, $F_r(11) = (T11 - T12) \times W$ $F_r(12) = (T9 - T10) \times W$, $F_r(13) = (T7 - T8) \times W$ $F_r(14) = (T5 - T6) \times W$, $F_r(15) = (T3 - T4) \times W$ $F_r(16) = (T1 - T2) \times V$ $F_r(N - i) = F_r(i)$ for $i = 1, 2, \dots, 15$
Imaginary Coefficients	Intermediate	$B_{131} = f_1 - f_{31}$, $B_{230} = f_2 - f_{32}$, $B_{329} = f_3 - f_{29}$, $B_{428} = f_4 - f_{28}$, $B_{527} = f_5 - f_{27}$, $B_{626} = f_6 - f_{26}$, $B_{725} = f_7 - f_{25}$, $B_{824} = f_8 - f_{24}$, $B_{923} = f_9 - f_{23}$, $B_{1022} = f_{10} - f_{22}$, $B_{1121} = f_{11} - f_{21}$, $B_{1220} = f_{12} - f_{20}$, $B_{1319} = f_{13} - f_{19}$, $B_{1418} = f_{14} - f_{18}$, $B_{1517} = f_{15} - f_{17}$ $G = 1024 \times B_{824}$ $G_{p1} = A_{131} + A_{1517}$, $G_{m1} = A_{131} - A_{1517}$ $G_{p2} = A_{329} + A_{1319}$, $G_{m2} = A_{329} - A_{1319}$ $G_{p3} = A_{527} + A_{1121}$, $G_{m3} = A_{527} - A_{1121}$ $G_{p4} = A_{725} + A_{923}$, $G_{m4} = A_{725} - A_{923}$ $G_{p5} = A_{230} + A_{1418}$, $G_{m5} = A_{230} - A_{1418}$ $G_{p6} = 724 \times (A_{428} + A_{1220})$, $G_{m6} = 1024 \times (A_{428} - A_{1220})$ $G_{p7} = A_{626} + A_{1022}$, $G_{m7} = A_{626} - A_{1022}$ $G_{p8} = G_{m1} + G_{m4}$, $G_{m8} = G_{m1} - G_{m4}$ $G_{p9} = G_{m2} + G_{m3}$, $G_{m9} = G_{m2} - G_{m3}$ $G_{p10} = 724 \times (G_{m5} + G_{m7})$, $G_{m10} = G_{m5} - G_{m7}$ $T1 = 200 \times G_{p1} + 569 \times G_{p2} + 851 \times G_{p3} + 1004 \times G_{p4}$ $T2 = 392 \times G_{p5} + G_{p6} + 946 \times G_{p7} + G_8$ $T3 = 392 \times G_{m8} + 946 \times G_{p9}$ $T4 = G_{p10} + G_{m6}$ $T5 = 569 \times G_{p1} + 1004 \times G_{p2} + 200 \times G_{p3} - 851 \times G_{p4}$ $T6 = 946 \times G_{p5} + G_{p6} - 392 \times G_{p7} - G_8$ $T7 = 724 \times (G_{m8} + G_{m9})$

Imaginary Coefficients	Intermediate	**$T8 = 1024 \times G_{m10}$** **$T9 = 851 \times G_{p1} + 200 \times G_{p2} - 1004 \times G_{p3} + 569 \times G_{p4}$** **$T10 = 946 \times G_{p5} - G_{p6} - 392 \times G_{p7} + G_8$** **$T11 = 946 \times G_{p8} - 392 \times G_{p9}$** **$T12 = G_{p10} - G_{m6}$** **$T13 = 1004 \times G_{p1} - 851 \times G_{p2} + 569 \times G_{p3} - 200 \times G_{p4}$** **$T14 = 392 \times G_{p5} - G_{p6} + 946 \times G_{p7} - G_8$** **$T15 = G_{m8} - G_{m9}$**
	Final	**$F_i(0) = 0$,** **$F_i(1) = -(T1 + T2) \times W$** **$F_i(2) = -(T3 + T4) \times W$,** **$F_i(3) = -(T5 + T6) \times W$** **$F_i(4) = -(T7 + T8) \times W$,** **$F_i(5) = -(T9 + T10) \times W$** **$F_i(6) = -(T11+T12) \times W$,** **$F_i(7) = -(T13+T14) \times W$** **$F_i(8) = -T15 \times V$,** **$F_i(9) = -(T13-T14) \times W$** **$F_i(10) = -(T11-T12) \times W$,** **$F_i(11) = -(T9-T10) \times W$** **$F_i(12) = -(T7-T8) \times W$,** **$F_i(13) = -(T5-T6) \times W$** **$F_i(14) = -(T3-T4) \times W$,** **$F_i(15) = -(T1-T2) \times W$** **$F_i(16) = 0$** **$F_i(N - i) = -F_i(i)$** **for $i = 1, 2, \ldots, 15$**

The audio signal is represented in frequency domain (i.e., Fourier space) by the real and imaginary coefficients. And, alternatively, they can be expressed in polar form, in other words instead of representing the signal in terms of $F_r()$ and $F_i()$, it could be represented by two other parameters, called magnitude, written as *Mag()*, and phase, written as *Phs()*, see equations(2.6) and (2.7). The magnitude and phase are pair-to-pair equivalents with the real and imaginary parts. For example, *Mag(0)* and *Phs(0)* are calculated by using only $F_r(0)$ and $F_i(0)$. Likewise, *Mag(7)* and *Phs(7)* are calculated by using only $F_r(7)$ and $F_i(7)$, and so forth.

The value of magnitude is always positive, because is it the output of the square root function, see equation (2.6). The phase angle is represented in degrees (ranging from 0 and 360 degrees).

The abovementioned symmetry feature could be applied on the phase and magnitude parameters see equations (2.12.c) and (2.12.d). Therefore, due to this characteristic, just the half-first coefficients of the magnitude and phase are calculated and the remaining coefficients are directly computed by using equations (2.12.c) and (2.12.d). This step is also, useful to reduce the number of computations and consequently, the time needed to perform the calculations. Code list (3.6) illustrates the calculation of the magnitude and the phase coefficients.

Code List (3.6): Construct Phase & Magnitude Values

Input:

F_r: array of real part coefficients.

F_i: array of imaginary part coefficients.

N: audio block size. *// i.e., 8, 12, 16, 24, 32*

Output:

Phs: Array of phase values

Mag: Array of magnitude values

Begin

Set Nh ← (N \ 2 - 1)

Set D2R ← $tan^{-1}(1) / 45$

For k=1 to Nh

Set Mag(k) ← Sqr($F_r(k) \times F_r(k) + F_i(k) \times F_i(k)$)

If $F_r(k) = 0$ then

If $F_i(k) > 0$ then Set Phs(k) ← 90

ElseIf $F_i(k) < 0$ then Set Phs(k) ← 270

Else Set Phs(k) ← 0

Else

Set Phs(k) ← $tan^{-1}(F_i(k) / F_r(k)) / D2R$

Continue

```
                Set Phs(k) ← tan^-1(Fi(k) / Fr(k)) / D2R
                If Fr(k) < 0 Then Set Phs(k) ← Phs(k) + 180
                Else If Phs(k) < 0 Then Set Phs(k) ← Phs(k) + 360
            End if
            Set Mag(N-k) ← Mag(k)
            Set Phs(N-k) ← 360 – Phs(k)
        End loop
End.
```

Once the secret data embedded in the phase coefficients, then the real and imaginary coefficients should be reconstructed from the magnitude and phase coefficients by using equations (2.8) and (2.9). The determined real and imaginary coefficients are passed through the *inverse discrete Fourier transform (IDFT)*; see equation (2.10), to reconstruct the audio stego signal. The reconstructed audio block must be similar to the original audio block. Due to the embedding process, the phase coefficients are changed, but these changes should not cause a subjective effect on the sound quality. Taking into consideration the fact that the changes in phase coefficients have less effect on the sound in comparison with the effect of changes in the magnitude coefficients. Code list (3.7) illustrates the implemented steps to determine the real and imaginary part coefficients from the magnitude and phase values. While, code list (3.8) shows the steps of the inverse discrete Fourier transform (IDFT).

Code List (3.7): Reconstruct Real & Imaginary Part

Input:

Phs : array of the phase values

Mag : array of the magnitude values

N : wave block size. *// i.e., 8, 12, 16, 24, 32*

Output:

F_r *: array of real part coefficients.*

F_i *: array of imaginary part coefficients.*

Begin

Set Nh ← (N \ 2-1)

Set D2R ← tan^{-1}(1) / 45

For k = 1 to Nh

Set $F_r(k) \leftarrow Mag(k) \times \cos(Phs(k) \times D2R)$

Set $F_i(k) \leftarrow Mag(k) \times \sin(Phs(k) \times D2R)$

Set $F_r(N-k) \leftarrow F_r(k)$, and Set $F_i(N-k) \leftarrow -F_i(k)$

End loop

End.

Code List (3.8): Inverse Discrete Fourier Transform

Input:

F_r*: array of real part coefficients.*

F_i*: array of imaginary part coefficients.*

C: table of cosine data.

S: table of sine data.

N: block size.

Output:

WavBlk : vector of reconstructed wave data of length N.

Begin

For j = 0 to N-1

Set Sum ← 0

For i = 0 to N-1

Set Sum ← Sum + $F_r(i) \times C(i , j) – F_i(i) \times S(i ,j)$

Set WavBlk(j) ← Sum ⁄ 1024

End

3.4.3 Quantization Process

Quantization is a process comes after the forward Fourier transform, it is applied on the phase coefficients. The type of the applied quantization was the uniform *scalar* quantization, where each sample is treated individually. The phase coefficients are *uniformly* quantized in order to establish suitable uniform slack spaces within their values, these slack space are utilized as hosting areas, to hide secret data. The gaps made between the quantized values of the phase coefficients should be enough to carry the added values of the secret data. The uniform scalar quantization was done by using the following equation:

$$Phsq(u) = Q \times round\left(\frac{Phs(u)}{Q}\right) \qquad (3.2)$$

Where:

Phs(u): is the u^{th} original phase coefficient.

Q: is the quantization step.

$Phs_q(u)$: is the corresponding u^{th} quantized coefficient.

u = 0, 1, … , N - 1

The values of the phase coefficients should lay within the range [0,360|, also the value of the quantization step must be within this range.

3.4.4 Hiding in Selected Blocks

In this research, two hiding methods were proposed for hiding a binary secret data in an audio cover. Both methods differ in the criteria of choosing the host audio blocks. These established hiding methods are:

A. Hiding based on voiced/unvoiced blocks detection (Hiding in voiced blocks (HVB)).
B. Hiding based on secret integrity check (Hiding in checked blocks (HCB)).

A. Hiding Based on Voiced/Unvoiced Blocks Detection

The strategy of this method is based on partitioning the audio cover data into small blocks, the size of each block is N, then each block is classified into voiced/unvoiced block. And as a final stage, the secret data are embedded into the voiced segments.

This method consists of two stages Hiding and Extraction; they will be described in following subsections:

1. The Hiding Stage

In this stage, the hiding of secret block is done on the voiced blocks, exclusively. So, the implementation of Fourier transform and quantization process (to prepare the slack space) are done on the voiced blocks. The embedding process is done by adding or subtracting a (Δ) value to the quantized phase coefficients. The application of addition or subtraction processes depends on the value of the secret bit (whether it is 0 or 1). *The value of (Δ) should be less than half the value of quantization step, in order to avoid the occurrence of a jump from certain quantization bin to the one of the adjacent (pervious or next) quantization bins.* To make the process of determining the suitable values of (Δ) more easy and consistent parameter called slack step ratio (*R*) is proposed, it is a ratio parameter define as follow:

$$R = \frac{\Delta}{Q} \qquad (3.3)$$

The value of *R* should be ($0 < R < 0.5$). In this research, the value of *R* was taken either (1/3) or (1/4). Both the values of (*Q*) and (*R*) are predefined by the user.

The quantity (Δ) is added to quantized phase (phs_q) if the secret data value (*Scrt*) is 1, or it is subtracted if (*Scrt*) is 0, as follows:

$$Phs_H(u) = \begin{cases} Phs_q(u) - \Delta & if\ Scrt(i) = 0 \\ Phs_q(u) + \Delta & if\ Scrt(i) = 1 \end{cases} \quad (3.4)$$

Where,

$Phs_q(u)$ is the u^{th} quantized phase coefficient in the block.

$Phs_H(u)$ is the u^{th} host phase coefficient in the block.

$Scrt(i)$ is the i^{th} secret bits.

u = 1, 2, …… , N/2 – 1

The above equation indicates that the embedding of secret data is done on the first half of the AC-coefficients, this is due to symmetry feature of the Fourier transform coefficients, see equation (2.12c). Where the second half of Fourier coefficients are related with the coefficients in the first half, and they cannot take different values. Therefore, the total secret bits, which will be hided in each block are (N/2-1). Code list (3.9) shows the implemented steps of this hiding method.

Code List (3.9): HVB Hiding Stage

Input:

***Cov**: The original samples of cover audio file of length **DataSize**.*

***Scrt**: An array of secret bits of size **ScrtSiz**.*

***N**: The block size.* *// N = 8, 12, 16, 24, 32*

Str**: Start block position of **OverInfo** bits vector in **Cov.

Ed**: Ending block position of **OverInfo** bits vector in **Cov.

Output:

***Steg**: Array of stego wave data of length **DataSize**.*

***Step 1:** Divide **Cov** into blocks of size **N** samples.*

***Step 2:**Check if (block position>=**Str**)**And**(block position<=**Ed**) then go to **step9**.*

***Step 3:** Check each block is it unvoiced block by applying code list (3.4) if it is unvoiced go to **step 8**.*

***Step 4:** Apply QFT on this unvoiced block to produce F_r and F_i, coefficients then construct **Phs** and **Mag** by using code list (3.6).*

***Step 5:** Quantize the **Phs** values by using equation (3.2).*

***Step 6:** Insert secret bits **Scrt** in the quantized phase coefficients by using equation (3.4).*

***Step 7:** Reconstruct F_r and F_i coefficients by applying code list (3.7) and pass the result through **IDFT** to construct the stego block, then round the values to the range [0..255].*

***Step 8:** put the produced block in **Steg** array, which represents the stego data.*

Step 9:** If (remain secret bits of **Scrt** And remain cover block) go to **Step 2.

***Step 10:** End.*

2. The Extraction Stage

The extraction stage is similar to that mentioned in hiding stage but it is arranged in reverse manner. Instead of embedding the secret data it is extracted. The extraction of the secret bits (either 1 or 0), is done according to the following equation:

$$Rscrt(i) = \begin{cases} 0 & If\ Phs_q(u) > Phs(u) \\ 1 & Otherwise \end{cases} \quad (3.5)$$

Where:

Phs(u): is the u^{th} phase coefficient of the stego audio block.

$Phs_q(u)$: is the u^{th} quantized phase coefficients.

Rscrt(i): is the i^{th} retrieved secret data.

u = 1, 2, … , N\2-1.

While code list (3.10) illustrates the main steps of the extraction stage.

Code List (3.10): HVB Extraction Stage

Input:

Steg*: Array of stego wave data of length* ***DataSize****.*

Str*: Start block position of* ***OverInfo*** *bits vector in* ***Steg.***

Ed*: Ending block position of* ***OverInfo*** *bits vector in* ***Steg.***

Output:

Scrt*: Secret bits of length* ***ScrtSiz****.*

Step 1: *Divide* ***Steg*** *into blocks of size* ***N*** *samples.*

Step 2: *Check if (block position>=****Str****)* ***And*** *(block position<=****Ed****) then go to* ***step7.***

Step 3: *Check each block whether it is voiced or unvoiced block, by applying code list (3.4), if it is unvoiced jump to* ***step 6****.*

Step 4: *Apply QFT on the (voiced block to produce* $\boldsymbol{F_r}$ *and* $\boldsymbol{F_i}$*, coefficients then construct* ***Phs*** *and* ***Mag,*** *by using code list (3.6).*

Step 5: *Quantize the* ***Phs*** *values by using equation (3.2) to produce* $\boldsymbol{Phs_q}$*.*

Step 6: *Retrieve secret bits* ***Scrt*** *from* $\boldsymbol{phs_q}$ *by using equation (3.5).*

Step 7: *If (retrieve secret bits <* ***ScrtSiz****)*

If remain blocks of ***Steg*** *then go to next block and jump to* ***step 2****.*

Else stop with error message indicating that stego file is corrupted

Step 8: End

This method has a disadvantage; not all the embedded secret bits are retrieved correctly, a little amount of secret bits may retrieved wrong. For this reason, another (second) hiding method was suggested to ensure the correct retrieval of all secret bits.

B. Hiding Based on Secret Integrity Check

The strategy of this method is based on partitioning the audio cover data into blocks with size N+3, where a set of three additional samples before each block (of N samples) is reserved to hold the value of a *flag* (either 0 or 1) to indicate whether the next block of N samples convey secret data or not. Therefore, the total size of each audio block is (N+3) samples instead of (N) samples.

This method could be considered better than the first method, because if offers perfect retrieval for the secret data. This hiding method consists of two stages: hiding and extraction.

1. The Hiding Stage

This stage uses the same processes that used in the previous hiding method, but with exception of voiced/unvoiced classification step. All the cover blocks are passed through QFT, quantization process, insertion equation (3.4), and finally IDFT process. In addition, to insure that all secret data will be retrieve correctly, the extraction of secret data is performed to check whether the retrieved secret data from the block are totally correct or not. If the retrieved data is correct then the cover block will be utilized as a host block, otherwise, it is ignored (left in its original from without any changes). Therefore, to do this check the steps of extraction process were added to this hiding stage, and it is started after IDFT. Therefore the involved processes of this hiding stage are: QFT, quantization, and the retrieval (equation 3.5). The retrieved secret data are

compared with the original secret bits, if the result is identical, then this block is classified as a host holding secret bits, otherwise it is left.

The first three samples of each block are used to hide the *flag* value, which in turn refers to the type of the block (i.e., is it a host block or not).

The least significant bit (LSB) encoding could be used to hide the flag-bit, so according to this technique least significant bit of the middle sample is set to zero when the block don't contains secret data, otherwise it is set to one when the block contains secret data (host block). Figure (3.2) illustrates the steps of hiding stage.

2. The Extraction Stage

This stage begins with partitioning the stego data into blocks, each has as size (N+3) samples. The first three samples are tested to know if the block contains secret data or not. In the case of LSB encoding, this test is done by checking if least significant bit of the middle sample, if it is zero then the block is considered empty (i.e., doesn't hold any secret data) otherwise it is considered as a host block. Code list (3.11) illustrated the steps of the extraction stage.

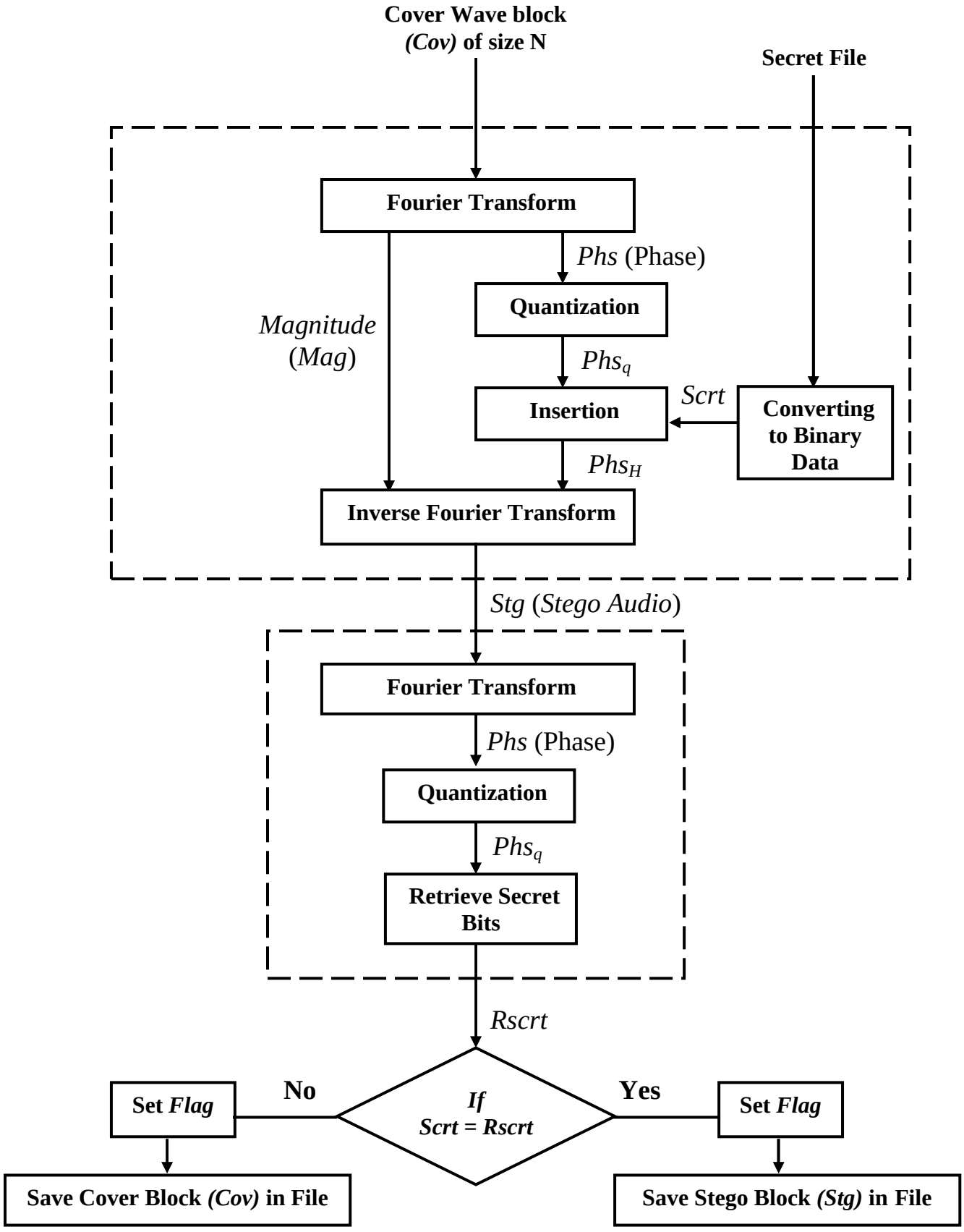


Fig (3.2) The layout of the hiding stage of HCB method

Code List (3.11): HCB Extraction Stage

Input:

Steg*: Array of stego wave data of length* ***DataSize.***

Str*: Start block position of* ***OverInfo*** *bits vector in* ***Steg.***

Ed*: Ending block position of* ***OverInfo*** *bits vector in* ***Steg.***

Output:

Scrt*: Secret Bits of length* ***ScrtSiz****.*

Step 1: *Divide* ***Steg*** *into blocks of size* ***N+3*** *samples (where the first three samples before each block represented* ***flag****).*

Step 2: *Check if (block position>=****Str****)* ***And*** *(block position<=****Ed****) then go to* ***step7.***

Step 3: *Check block* ***flag*** *if indicate to empty block (there are not secret bits in remain samples block) go to* ***step 7****.*

Step 4: *Apply QFT on the block of size N to produce* F_r *and* F_i*, coefficients then construct* ***Phs*** *and* ***Mag*** *by using code list (3.5).*

Step 5: *Quantize the* ***Phs*** *values by using equation (3.2) to produce* Phs_q*.*

Step 6: *Retrieve secret bits* ***Scrt*** *from* phs_q *by using equation (3.5).*

Step 7: *If (retrieve secret bits <* ***ScrtSiz****)*

If remain blocks of ***Steg*** *then go to next block and jump to* ***step 2****.*

Else stop with error message indicating that stego file is corrupted

Step 8: End

3.5 Overhead Information

Since the proposed hiding system uses many control parameters (like, block size, quantization step, slack ratio, …) which can be set by the user of the system. Therefore, in order to extract the secret data from the stego-object these control parameters should be fed to extractor. So, these parameters should be passed either through a secure channel to the receiver (extractor), or it should be embedded into the stego-file by using

the same hiding method but with a fixed-predefined control hiding parameters.

In the remaining part of this section a list of the parameters that required during the stage of extraction is given. These necessary parameters are considered as overhead information, and could be treated as a stego keys of the hiding system. In this research, the overhead information are coded to binary code words, where the number of bits used to represent each parameter depends on the nature and range of values of the parameters. Table (3.6) illustrates the number of bits assigned to each parameter.

Table (3.6) the overhead information description

Parameters	No. of Bits	Details
MthdNo	**1**	***To represent the type of hiding method (if 0 mean first method else second method).***
N	**3**	***To encode the size (in samples) of audio cover blocks.***
ScrtSize	**25**	***To represent the secret data size (in bytes).***
FilExtn	**24**	***The secret file name extension, the maximum file extension is three characters.***
Q	**7**	***To represent the quantization step (Q).***
Thr	**4**	***To represent the threshold value (Thr), this parameter is necessary when the type of hiding method is used.***

All the binary sequences were assembled into a single vector (called *OverInfo*); the size of this vector is 64 bits (for the first type of the hiding methods) or 60 bits for the second type of hiding method. If the user decided to embed these overhead information in the stego audio data they should be embedded (and extracted) before the stage of embedding (and extraction) of the secret data. In addition, their place in the stego-object file could be variant, but should be well known to both sides (encoder/transmitter, and decoder/receiver). Because these information

are very important in the extraction stage, so all the embedded secret bits should be retrieved correctly. Therefore, these information should be hided by using second method, which is based on secret integrity check. The binary vector (*OverInfo*) will be hiding in cover audio file at certain location, this location could be considered as a public key between sender and receiver. Code list (3.12) shows the implemented steps for hiding *OverInfo* vector in cover audio data. While code list (3.13) shows the implemented steps for extraction the overhead information vector.

Code List (3.12): Hiding the OverInfo vector

Input:

OverInfo*: A vector contains binary bits of the overhead information, it has a size (64) bits for HVB hiding method, and (60) bits for HCB hiding method.*

Cov*: Original samples of cover audio file of length* ***DataSize****.*

Output:

Steg*: An array of stego wave data.*

Str*: Start block position of* ***OverInfo*** *bits vector in* ***Steg.***

Ed*: Ending block position of* ***OverInfo*** *bits vector in* ***Steg.***

Step 1: *Divide* ***Cov*** *into blocks of size* ***(N+3)*** *samples (where the first three samples in each used as a* ***flag*** *status).*

Step 2: *Divide these blocks into two equal groups.*

Step 3: *Start from the first block of the second group by setting* ***Str*** *to this position and setting* ***Ed*** ← ***Str****.*

Step 4: *Apply on the block of position* ***Ed*** *second hiding method* ***(HCB)****, figure (3.2)), (by insert N/2-1 bits of* ***OverInfo****) .*

Step 5: *Saved block of size (N+3) in* ***Steg*** *array, and*

Step 6: *increase* ***Ed*** *by N+3*

Step 7: *If (OverInfo bits not finished) then*

*Check If (****Ed*** *>* ***DataSize****) then exit Else go to* ***step 4.***

Step 8: End

Code List (3.13): Extraction the OverInfo vector

Input:

Steg**: Array of stego wave data of length **DataSize.

Str**: Start position of OverInfo bits vector in **Steg.

Ed**: End position of OverInfo bits vector in **Steg.

Output:

***OverInfo**: A vector contains binary bits of the overhead information, it has a size (64) bits for HVB hiding method, and (60) bits for HCB hiding method.*

***Step 1:** Divide **Steg** into blocks of size **N+3** samples (where the first three samples in each block used to hide **flag**).*

***Step 2:** Check **flag** of **Str** block position*

*If this block consider host then extract **OverInfo** bits by using HCB extracting stage, code list (3.10).*

***Step 3:** Increased **Str** by N+3.*

***Step 4:** If **OverInfo** bits not complete (60 bits for HCB method, and 64 for HVB method) then*

*If **Str** < **Ed** Then go to **step 2***

Else Exit //error in stego file

***Step 5:** End*

3.6 Construction of Stego Audio File

After the completion of the stage of embedding secret data in the cover audio media, then the stego wave file should be constructed. It consists of the header section and the data section of stego audio samples. The original audio samples may be read as 8 or 16 bits/sample, but after hiding they will be of resolution 8 bits/sample. This conversion is very useful in the implementation phase of the proposed hiding method, because it is better to hide in audio media with a resolution 8 bits/sample, because their quality are less than the 16 bits audio samples, and can convey some sort of distortion without making suspicion, so the

distortion due to hiding process will be not easy to recognize and discriminated from the noise which naturally associated the low quality audio samples. Table (3.7) illustrates the contents of the header section, for more information about each field in the header; see tables (2.1), (2.2) and (2.3).

Table (3.7) The Header Structure

	offset	***Size (Byte)***	***Field name***	***Description***
The "RIFF" chunk	**0**	**4**	**Chunk ID**	**"RIFF"**
	4	**4**	**Chunk Size**	**File size = 36 bytes+ *DataSize***
	8	**4**	**Format**	**"WAVE"**
The "fmt " sub-chunck	**12**	**4**	**Sub-Chunk1 ID**	***"fmt "***
	16	**4**	**Sub-Chunk1 Size**	**Always equal to 16 bytes**
	20	**2**	**Audio Format**	**Audio format = 1 (PCM)**
	22	**2**	**Channel Numbers**	**Channels = 1 (Mono)**
	24	**4**	**Sample Rate**	**Sample Rate**
	28	**4**	**Byte Rate**	**= Sample Rate because it equal to (Sample Rate×Channel No.×Bit Per Sample / 8)**
	32	**2**	**Block Align**	**1**
	34	**1**	**BitsPerSample**	**8**
The "data" sub-chunk	**36**	**4**	**Sub-Chunk2 ID**	**"data"**
	40	**4**	**Sub-Chunk2 Size**	**Data Size**
	44	**-**	**Data**	

3.7 Extraction Module

As mentioned in previous section, the extraction stage consists of similar stages to that used in hiding modules. As a first step in extraction module the vector (*OvrInfo*), which contains the overhead information needed to perform the correct extraction, should be retrieved.

Then, the stego wave file is partitioned into blocks have same size like the size of blocks used in the hiding module. Figure (3.3) illustrates the structure of extraction stages, the extraction module steps can be summarized as follows:

1. Load stego audio data from a wave file.
2. Extract overhead information (*OvrInfo*).
3. Determine the hiding method type from the vector (*OvrInfo*).
4. Partition the stego audio samples into blocks of size (N) samples if the hiding method in based on voiced/unvoiced blocks classification method (HVB), or into blocks of (N+3) samples if the applied hiding method was the second one (HCB) (i.e., based on secret data integrity). The value of N should be determined from the vector (*OvrInfo*).
5. Test each block to know if it is hosting secret bits or not, this action also depends on type of the hiding method, either by voiced/ unvoiced check or flag check.
6. **Fourier transform:** each stego block is transformed to frequency domain by using quick Fourier transform method to get the real and imaginary coefficients, and then the phase coefficients are calculated.
7. **Phase quantization:** the phase coefficients are quantized by applying the quantization equation (3.2), to produce quantized phase (Phs_q) coefficients.

8. **Secret data extraction:** In this step the secret bits are extracted from the stego block, by using equation (3.5). The results from this step will assembled in one vector represent the secret data.
9. **Secret Data construction:** after extraction of secret bits. Then each 8-bits are merged to construct a byte, then all the constructed bytes are put in a file whose type could be defined from the file name extension which is already exists as a part of the overhead information.

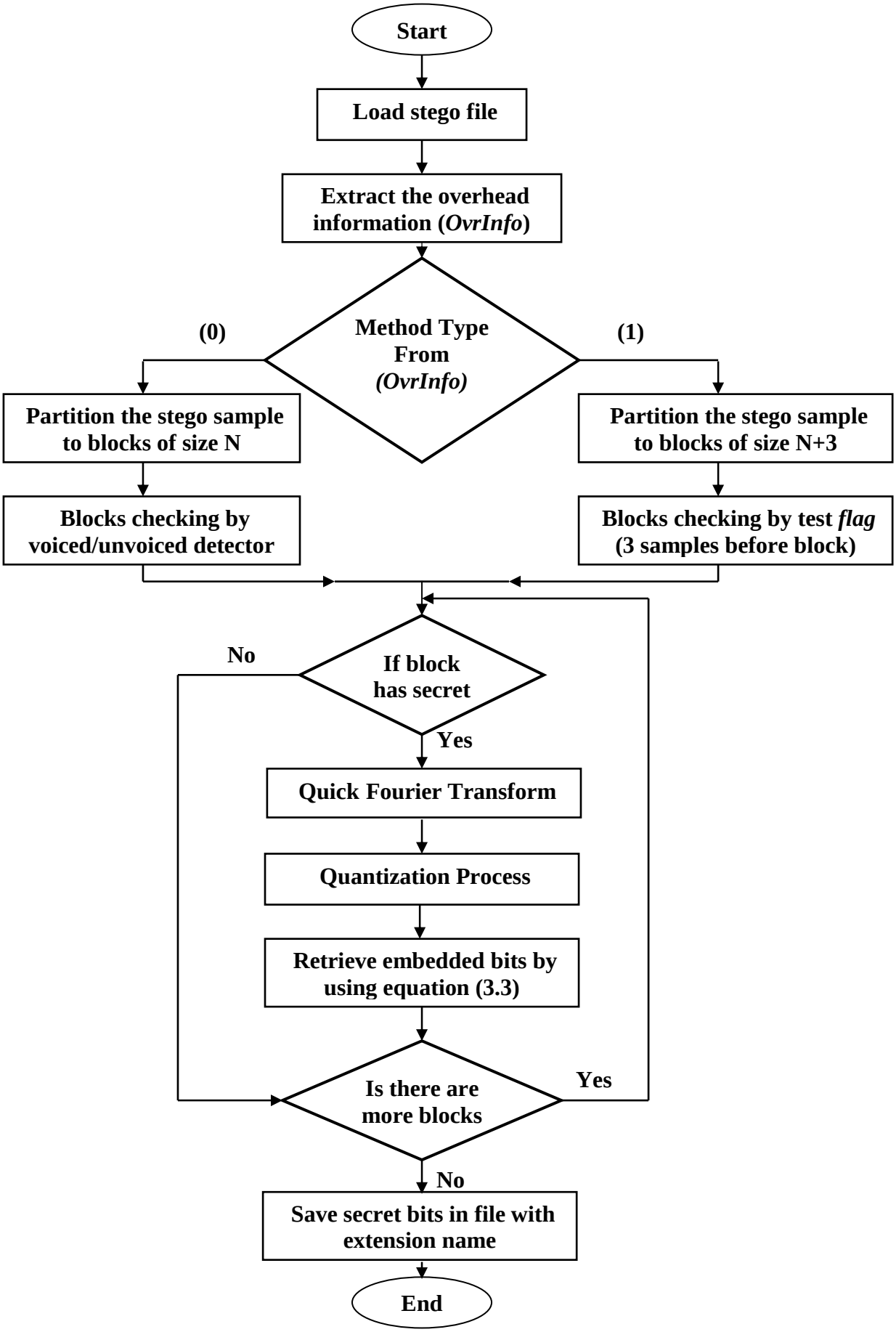


Fig (3.3) The flowchart of the extraction module

Chapter Four
Performance Test Results

4.1 Introduction

This Chapter is dedicated to present the results of the conducted tests to evaluate the performance of the proposed hiding methods. The measures used in these tests are the hiding rate, the percentage ratio of the correct retrieved secret bits, and some fidelity measures. The performance tests were performed on the proposed two hiding methods by using some selected audio files as test material. The effects of the involved control parameters have been studied; specifically their effects on the hiding rate and on the quality of stego data.

4.2 The Test Measures

The measures used in this project to assess the performance of the considered two hiding methods are the following:

A. The Fidelity Criteria

These measures have been used to represent the level of the overall error caused in stego-cover due to embedding of secret data. The adopted Fidelity measures could be defined as follows [Sal00]:

$$MSE = \frac{1}{M}\sum_{i=1}^{M}(s_i - s_i')^2 \tag{4.1}$$

$$SNR = 10\log_{10}\left(\frac{\sum_{i=1}^{M} s_i^2}{\sum_{i=1}^{M}(s_i - s_i')^2}\right) \tag{4.2}$$

$$PSNR = 10\log_{10}\left(\frac{255^2}{MSE}\right) \tag{4.3}$$

Where, s_i is the i^{th} sample in the audio cover data.

s_i' is the corresponding i^{th} sample in the audio stego-cover data.

M is the total number of audio samples.

B. Hiding Rate (HR)

This parameter is determined to assess the capability of the tested hiding method to embed secret information in the cover object. Mathematically, hiding rate is defined as follows:

$$HR = \frac{S_t}{8 \times M} \times \%100 \qquad (4.4)$$

Where, S_t is the maximum possible number of inserted bits in the cover media.

M is the size of cover media in bytes.

C. Ratio of Wrong Retrieved Secret Bits (WR)

This parameter is determined to describe the capability of the tested hiding method to preserve the accuracy of the embedded secret data. It is defined as the ratio of the number of wrong secret bits (S_w) extracted from the stego object relative to the total number (S_t) of embedded secret bits:

$$ER = \frac{S_w}{S_t} \times \%100 \qquad (4.5)$$

4.3 The Test Samples

Some audio files have been selected as cover media to investigate the effects of the involved parameter in both hiding methods. Table (4.1) shows a list of the selected cover files, while Table (4.2) shows a list of files whose contents are used as test material.

Table (4.1) The list of wave files used as cover media

File Name	File Size	Channel Type	Sample Resolution (bits)	Sampling Rate (kHz)
Sng1	545 KB	Stero	16	11 .00
Music1	99.1 KB	Mono	8	8
Sng2	268 KB	Stero	8	11 .00
Music2	260 KB	Mono	8	11 .00
Music3	248 KB	Mono	8	11 .00
Sng3	118 KB	Mono	8	8 .00
Spch1	856 KB	Mono	8	16 .00
Sng4	117 KB	Stero	8	11 .00
Spch2	146 KB	Mono	8	11 .00
Sng5	350 KB	Stero	8	11 .00
Music4	497 KB	Mono	8	22 .00
Sng6	633 KB	Stero	16	8 .00
Music5	752 KB	Mono	8	22 .00
Music6	395 KB	Mono	16	11 .00

Table (4.2) The list of files whose contents used as secret material

File Name	File Size (KByte)	File Type
DocSec	26	DOC
WavSec	236	WAV
TxtSec	13.1	TXT
ImgSec	12.3	JPG

4.4 Quick Fourier Transform Tests

The quick Fourier transform (QFT) technique was used to speed up the computation of the discrete Fourier transform (DFT), taking into consideration that some approximations were taken place to convert the float arithmetic operations to integer operations which are executed faster. Such approximations cause intrinsic error in QFT. This error was tested by making comparisons between the original audio data and its version after passing through QFT (forward and inverse). The conducted test had been applied on an audio array (its size is 780000 bytes), this array was partitioned into blocks whose size is N (8, 12, 16, 24, or 32) samples. Table (4.3) shows the obtained test results.

Table (4.3) Comparison between traditional DFT and QFT, and the error produced by QFT

Block Size	Fourier Transform Method	Computation Time per Block (μsec)		No. of Addition Operations	No. of Multiplication Operations
		With Check*	Without Check*		
8	DFT	5.289	1.769	112	128
	QFT	0.962	0.481	20	12
12	DFT	12.019	5.529	264	288
	QFT	1.202	0.481	38	24
16	DFT	19.872	8.654	480	512
	QFT	2.244	0.962	58	32
24	DFT	42.308	21.635	1104	1152
	QFT	3.846	1.923	104	56
32	DFT	69.239	32.051	1984	2048
	QFT	4.481	2.564	172	100

*****_Visual basic programming has many kinds of advanced checks optimizations (like, integer overflow, floating point error, array bounded, un-rounded floating point operations)._

4.5 Performance of The Hiding Methods

The followed strategy to investigate the performance of the two hiding methods was based on studying the roll of the control parameters on the hiding results. The results of the various conducted tests have indicated that the type of cover audio file has significant effects on the performance parameters (like, hiding rate, correct retrieval ratio, and the quality of stego audio file). Therefore, the choice of audio cover file should be handled carefully, for example the clear audio data is not recommended because hiding in such media will cause the appearance of noticeable distortion. It was noticed that the results of the tests vary from cover to another, but the behavior patterns of the effects of control parameters are nearly similar. In the following paragraphs, the test results for both hiding methods are presented and discussed.

4.5.1 HVB Hiding Method Tests Results

In this method, the control parameters that have significant effects on the hiding rate and the ratio of the correctly retrieved secret bits; are the threshold value, quantization step, slack ratio and the block size. In the following sub sections the effects of the above control parameters on the performance of the hiding method are given.

A. Threshold Value (Thr)

The decision of voiced/unvoiced sound classification depends on the value of threshold (Thr). If the average energy of the tested audio block is higher than threshold (Thr) then the block is classified (discriminated) as voiced block, otherwise it is classified as unvoiced. Then, the voiced blocks are used as host area for hiding the secret data. Large threshold value ensure the correct retrieval of secret data is larger, but in such case a small number of cover blocks will be used to host secret data, so hiding rate is decreased and notice decreased in noise signal because avoided the unvoiced blocks. In the conducted tests to investigate the effect of threshold, the values of other constant parameters were taken as follows: α=5; β=1.25 for voiced block; while β=0.75 for unvoiced block. Table (4.4) illustrates the effect of threshold (Thr), where HVB hiding method is used to insert the data of (*WavSec*) file in (*music1*) audio cover file (see tables 4.1 and 4.2).

Table (4.4) The effect of the threshold (Thr) on the performance of HVB hiding method, (N = 8; Q = 30; R = 1/4).

Thr	Hiding Rate (%)	Wrong extracted Bits (%)	Fidelity		
			MSE	SNR	PSNR
2	4.63	2.32	6.079	34.43	40.29
4	4.59	2.18	6.087	34.43	40.287
8	4.09	1.94	6.099	34.42	40.28
11	3.31	1.51	5.829	34.62	40.47
12	3.02	1.39	5.582	34.81	40.66
14	2.44	1.34	5.003	35.28	41.139
16	1.97	1.08	4.447	35.79	41.65
20	1.31	1.28	3.181	37.25	43.11

B. Quantization Step (Q)

Quantization process provides suitable slacked spaces in phase values to hide secret data. The main parameter used to manage the amount of slack space is the quantization step (*Q*). If the value of *Q* is increased then the slack space used for hiding increases and the error in retrieved bits are decreased, but at the same time it causes an increase in the distortion level of the stego-cover signal. Table (4.5) presents the effect of quantization step (*Q*) on the hiding method, the contents of the (*DocSec*) file was taken as a secret data, and (*Music2*) audio file as a cover media (see tables 4.1 and 4.2). Hiding ratio in table (4.5) doesn't show any changes, and its value was *(1.42)*, because *Thr* parameter is effected first on the hiding rate by determined which blocks will be consider host or not.

Table (4.5) The effect of the quantization step (Q) on the performance of HVB method, (N =8; Thr = 20; R = 1/4).

Q	Wrong extracted Bits (%)	Fidelity		
		MSE	SNR	PSNR
8	13.38	0.709	43.61	49.63
12	8.29	1.119	41.62	47.64
18	4.79	2.031	39.03	45.05
24	3.15	3.377	36.83	42.85
30	2.39	5.048	35.08	41.11
35	2.33	6.773	33.80	39.82
40	1.40	8.514	32.81	38.83

C. Slack Ratio (R)

This parameter specifies the amount of modulation within the slack space to encode the secret data. The value of *R* should be less than 0.5 to avoid the occurrence of shifting of the modulated cover coefficients from a quantization bin to one of the surrounding bins. Therefore, the effect of *R* was tested for the two cases: (1/3) and (1/4). Table (4.6) shows the effect of slack ratio *R*, the secret data was taken from (*ImgSec*) file and embedded in

(*Sng2*) audio file. Hiding ratio in table (4.6) also doesn't show any changes, and its value was *(1.91)*.

Table (4.6) The effect of the slack ratio (Q) on the performance of HVB method, (N =8; Thr = 20 Q = 30).

R	Wrong extracted Bits (%)	Fidelity		
		MSE	SNR	PSNR
1/3	2.06	6.267	34.26	40.16
1/4	1.67	4.932	35.30	41.20

D. Block Size (N)

Table (4.7) shows the effect of the block size (*N*), will notice, numbers of inserting bits are increased when value of (N) become large, but in the same time the distortion rate is increased because number of change sample in cover increased by embedding. The secret data was the contents of the (*Sng1*) file and the (*TxtSec*) audio file was taken as cover media.

Table (4.7) The effect of the block size (N) on the performance of HVB method, (Thr = 20; Q = 30; R = 1/4).

Block Size (N)	Hiding Rate (%)	Wrong extracted Bits (%)	Fidelity		
			MSE	SNR	PSNR
8	1.75	1.64	5.483	34.88	40.74
12	2.05	1.77	7.969	33.26	39.12
16	2.25	1.79	9.961	32.29	38.15
24	2.51	2.04	11.496	31.67	37.53
32	2.64	2.29	13.046	31.12	36.98

4.5.2 HCB Hiding Method Tests Results

The control parameters that affect the performance of this method are the quantization step (*Q*), slack ratio *(R)* and the block size (*N*). They affect, only, the hiding rate and quality of the audio file, because all the embedded bits are retrieved correctly. The problem faced this method is in its probable selection of some unvoiced blocks as cover area, taking into consideration that hiding in unvoiced audio blocks may cause some noticeable distortions.

To avoid the occurrence of this problem the audio cover file must be chosen carefully, such that the chosen audio file should not have long unvoiced segments. In the following subsections, the effects of some control parameters on the system performance are explored.

A. Quantization Step (Q)

Table (4.8) shows the effect of quantization step (*Q*) on the performance of HCB method, notice in this method, the value of (Q) is effected on the hiding rate because it consider the first parameter that determine which will be the host blocks or not. The contents of the (*DocSec*) file were used as secret data and (*Music2*) file used as a cover media.

Table (4.8) The effect of quantization step (Q) on the performance of HCB method, (N =8; R = 1/4).

Q	Hiding Rate (%)	Fidelity		
		MSE	SNR	PSNR
8	1.51	0.445	45.63	51.65
12	1.96	0.854	42.79	48.82
18	2.37	1.828	39.49	45.51
24	2.65	3.191	37.07	43.09
30	2.79	4.960	35.15	41.18
35	2.91	6.585	33.92	39.95
40	3.01	8.810	32.66	38.68

B. Slack Ratio (R)

Table (4.9) shows the effect of slack ratio (*R*), where the contents of (*ImgSec*) file have been used as secret data, and (*Sng2*) audio file as cover media.

Table (4.9) The effect of slack ratio (R) on the performance of HCB method, (N =8; Q = 30).

R	Hiding Rate (%)	Fidelity		
		MSE	SNR	PSNR
1/3	3.14	6.485	34.11	40.01
1/4	3.21	5.169	35.09	40.98

C. Block Size (N)

The effect of this control parameter is illustrated in Table (4.10), the contents of the file (*Sng1*) was taken as secret data, while the audio file (*TxtSec*) was used as cover media. The best hiding rate result find when *N=16* but not best in the fidelity criteria.

Table (4.10) The effect of block size (N) on the performance of HCB method, (Q = 30; R = 1/4).

Block Size (N)	Hiding Rate (%)	Fidelity		
		MSE	SNR	PSNR
8	2.99	5.027	35.26	41.12
12	3.41	8.431	33.01	38.87
16	3.49	10.574	32.03	37.89
24	3.39	11.260	31.76	37.62
32	3.06	13.437	30.99	36.85

4.6 Performance Comparison

In this section, a comparison between the hiding results obtained by applying the two considered methods (i.e., HVB and HCB) is given. Table (4.11) lists some of the best results obtained by using different cover and secret files. Also, the values of the involved control parameters where varied to explore the difference between the performance of the two hiding methods. It is obvious that the hiding rate and the distortion level change between the two methods depending on the audio cover obtained by using hiding based on voiced blocks method (HVB) and hiding based on secret integrity check method (HCB).

Table (4.11) The results of some test examples obtained by applying HVB and HCB hiding methods.

Cover	Secret	N	R	Q	Thr	Hiding Method	Hiding Rate (%)	Wrong extracted Bits (%)	Fidelity		
									MSE	SNR	PSNR
Music3	TxtSec	12	1/4	40	12	HVB	3.34	1.46	8.457	32.91	38.86
					----	HCB	3.61	----	7.319	33.54	39.49
Sng3	DocSec	16	1/3	24	16	HVB	3.85	6.40	14.170	30.75	36.62
					----	HCB	2.72	----	10.696	31.97	37.839
Spch1	DocSec	8	1/3	35	14	HVB	0.37	3.83	2.169	38.97	44.77
					----	HCB	1.84	----	2.564	38.24	44.04
Sng4	ImgSec	24	1/4	18	20	HVB	4.89	5.59	12.121	31.48	37.30
					----	HCB	2.77	----	12.085	31.49	37.31
Spch2	WavSec	12	1/4	30	11	HVB	3.51	3.73	19.255	29.47	35.29
					----	HCB	2.79	----	14.439	30.76	36.54
Sng5	WavSec	16	1/3	40	16	HVB	1.88	5.84	16.248	30.08	36.02
					----	HCB	2.40	----	14.179	30.67	36.61
Music4	WavSec	32	1/3	35	12	HVB	3.97	9.17	10.850	31.83	37.78
					----	HCB	1.35	----	3.572	36.66	42.60
Sng6	ImgSec	24	1/4	30	8	HVB	5.22	5.59	13.102	31.08	36.96
					----	HCB	2.81	----	10.133	32.19	38.07
Music5	DocSec	8	1/4	24	14	HVB	2.70	5.48	2.403	38.51	44.32
					----	HCB	2.69	----	1.832	39.68	45.50
Music6	TxtSec	12	1/3	30	11	HVB	4.01	2.79	14.11	30.74	36.64
					----	HCB	3.52	----	11.025	31.81	37.71

Chapter Five
Conclusions and Suggestions

5.1 Conclusions

From the test results listed in previous chapter the following remarks were derived:

1. Using quick Fourier transform (QFT) method is useful to reduce the computational load of the transform (by reducing number of mathematical operations). The main difficulty face the implementation of QFT lay in its long equations (i.e., coding complexity), which become very hard to manage when block size (N) increased. Therefore, this technique is hard to apply for large block size.
2. Hiding in voiced block sample is more suitable to avoid noise occurrence, which is more probably happen when unvoiced blocks are used as host area.
3. Although HVB method offer better hiding rate than HCB, but not all the secret bits are retrieved correctly. For this reason, this method is not suitable for hiding secret data whose integrity is very important requirement.
4. Large threshold value provide more power in cover audio signal by avoiding unvoiced blocks and increased correct retrieved bits, but decreased in hiding rate.
5. Large quantization step provides more slacked space to hide secret bits but causes more distortion in cover audio.
6. The HCB hiding method suffers from the appearance of little distortion due to hiding the integrity check flags put in some unvoiced blocks, some noticeable noise may appear in these areas.

To avoid this case the selected audio cover file should convey little unvoiced audio blocks.

5.2 Suggestions

As mentioned in some conclusion remarks that the developed hiding methods need some enhancements to improve their performance and to override some of their weak aspects. Therefore, these methods need further development in future, and in the following some suggestions are derived as future work developments:

1. Develop the system to be capable to handle stereo audio file as a cover media.
2. Develop the system to use another audio file formats like (MP3, DPCM, …)
3. Develop QFT method to be capable to handle large blocks (N >32).
4. Design and compute inverse quick Fourier transform method in manner like forward quick Fourier transform to get more reduction in computation time.
5. Combine the criteria used in both hiding methods to get better performance. This means the implementation of both voiced/unvoiced check and integrity check at the same time.
6. Develop the proposed hiding method to use image files as cover media instead audio cover, but the step of voiced/unvoiced checking should be replaced by contrast checking.

References

[AP98] Anderson, R. J., and Petitcolas, F. A., "On the Limits of Steganography", IEEE Journal of selected areas in communications, 16(4), pp 474-481, May, 1998.

[Ben96] Bender, W., "Techniques for Data Hiding", IBM system journal; vol. 35, no. 3-4, pp. 1-10, 1996.
http://www.research.ibm.com/journal/sj/353/section/bendeaut.html#bender

[Bor01] Bourke, P., "WAVE sound file format", November, 2001.
http://local.wasp.uwa.edu.au/~pbourke/dataformats/wave

[CF00] Cacciaguerra, S., and Ferretti, S., *"Data Hiding: Steganography and Copyright Marking"*, Paper, Department of computer science, Bologna University, Italy, 2000.

[Cha02] Chandramouli, R., "A Mathematical Approach to Steganalysis", multimedia system, networking and communications (MsyNc) lap, department of electrical and computer engineering, stevens institute of technology, California, January,2002.
http://www.ece.stevens-tech.edu/~msync

[Has01] Hass, J., "Principles of Digital Audio", html paper, Indiana University School of Music, center of electronic and computer music, November, 2001.
http://www.indiana.edu/~emusic/digital_audio.html

[Hei05] Heiman, D, "Acoustics and Fourier Transform", pdf paper, advanced physic lab-1, Northeastern University, July, 2005.

[JDJ00] Johnson, N. F., Duric, Z., and Jajodia. S., "Information Hiding: Steganography and Watermarking – Attacks and countermeasures", Kluwer Academic Publishers, 2000.

[JJ98-1] Johnson, N. F., and Jajodia, S., "Steganalysis of Image Created Using Current Steganography Software", Workshop on

information hiding proceeding, Center for secure information system, lecture notes in computer science, vol. 1525, Geoge Mason University, USA, 1998.
http://isse.gmu.edu/~csis

[JJ98-2] Johnson, N. F., and Jajodia, S., "Steganalysis: The Investigation of Hidden Information", article, IEEE information technology conference, New York-USA, 1998.

[KP00] Katzenbeisscr, S., and Petitcolas, F. A., "Information Hiding Techniques for Steganography and Digital Watermarking", Artech House, Boston-London, 2000.

[Lan99] Lane, D. E., "Video in Video Data Hiding", pdf paper, Department of Electrical and Computer Engineering, California University, USA, 1999.

[Maj04] Majeed, M. N, "Sound Hiding and Extraction Using DCT", M.Sc.Thesis, computer science department, college of science, Al-Nahrain University, Baghdad-Iraq, 2004.

[Mun04] Muntaha, A. J., "Image in Image Steganography Using Fourier Transform Coding", M.Sc. Thesis, applied science department, Al-Rasheed college of Engineering and Science, University of Technology, Baghdad-Iraq, 2004.

[MSY99] McClellan, J. H., Schafer, R. W., Yoder, M. A., "DSP First A multimedia Approach", Prentice-Hall, USA, 1999, pp. 24-334.

[PK01] Polpitiyg, A. D., and Kahn, W. J., "Information Hiding in Audio File with Encryption", Data security – project, Washington University, USA, Version 1.0, November, 2001.

[PWJ99] Petrovic, R., Winograd, J. M., Jemili, J., and Metois, E., " DATA HIDING WITHIN AUDIO SIGNALS", pdf paper, Electronics and Energetics, vol. 12, No. 2, pp. 103-122, 1999.

[Rob01] Robin, M., "Audio Sampling", html, December, 2001. http://broadcastengineering.com/aps/measure/broadcasting_audio_sampling/index.html#top

[Sal00] Salomon, D., "Data Compression The Complete Reference", Addision Wesley Company, Second Edition, 2000.

[Smi99] Smith, S. W., "Digital Signal Processing", California technical publishing, USA, 2nd edition, 1999.

[Tim98] Tim, K., "A Programmer's Guide to Sound", Addison-Wesley developer Press, 1998.

[Umb98] Umbaugh, S. E., "Computer Vision and Image Processing: A Practical Approach using CVIP Tools", Prentice Hall, USA, pp:268 and 246, 1998.

[Wit04] Witwit, A. M, "Audio in Audio Steganography Using Wavelet Transform", M.Sc. Thesis, computer science department, college of science, Baghdad University, Baghdad-Iraq, 2004.

[Web03] Weber, T. J, "Wave PCM Sound File Format", 2003. **http://www.ora.com/centers/gff/formats/micriff/index.html**

[Yas03] Yasmeen, I. F., "Text in Audio Watermarking System", M.Sc. Thesis, computer science department, college of science, Al-Nahrain University, Baghdad-Iraq, 2003.

[Zol98] ZolZer, U., "Digital Audio Signal Processing", John Wiley and Sons Ltd, England, reprinted 1998.

الخلاصة

الكتابة الخفية (Steganography) تعتَبرُ أحد الطرقِ كثيرة الإستعمالِ لإخفاء المعلومات وذلك بإخفاء بيانات سريةَ في غطاء رقمي بدون شك واضحِ.

هذا العملِ يركز على دراسة طريقتين لإخفاء أي نوعية من البيانات السرية في الإشارةَ الصوتيةَ. في البدء الإشارة السمعية تحول إلى (frequency domain) بإستعمال تسريع (Fourier Transform) التي تعتمد على تقليل الوقت وعدد العمليات الرياضية.

هناك طريقتان للأخفاء صممتا لتَضمين البتات السرية في معاملات الطور (Phase Domain) للإشارة السمعية. الطريقة الأولى، تسمى الإخفاء في الأجزاء الجهورية (HVB)، تتضمن إخفاء البتات السرية في الأجزاء الجهورية للبيانات السمعية. بينما الطريقة الثانية تسمى الإخفاء في الأجزاءِ المدققة (HCB)، حيث تتضمن إخفاء البتات السرية في الأجزاء المسموعة التي اجتازت إختبارات سلامة البتات المسترجعة بنجاح. هذه الطريقتان تستخدمان الملفات الموجية المسموعة ذات القناة الواحدة أو القناتين (mono and stereo).

وقد تم اختبار أداء طرقِ الإخفاء المُقتَرحة بإستعمال مقاييس الدقة خطأ المربع المتوسط (MSE)، نسبة ضوضاء الإشارة (SNR)، نسبة ضوضاء قمة الإشارة (PSNR) لقياس نسبة الخطأ، وتم حساب نسبة الإخفاء لتَقييم قوَّةِ الإخفاء لكل طريقة. كذلك تأثيرات بَعْض عوامل السيطرة على أداءِ النظامَ قد تم اختبارها لمُسَاعَدَة المستخدمِ لإختيار قِيَم عوامل النظام بشكل صحيح.

تُشيرُ نَتائِجُ الإختبار بأنّ طريقةَ (HVB) تعطينا قوة إخفاء أكثر مِنْ طريقةِ (HCB)، لكن لَيستْ كلّ البتات السرية المخفية يُمْكِنُ أَنْ تُسترجعَ بشكل صحيح عند أستعمال (HVB).

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة النهرين
كلية العلوم

الإخفاء بمعدل واطئ في بيانات الصوت بأستخدام مجال الطور

رسالة

مقدمة إلى كلية العلوم في جامعة النهرين

كجزء من متطلبات نيل شهادة الماجستير في علوم الحاسوب

من قبل

هبة محمد سليم الكواز

(بكالوريوس ٢٠٠٣)

المشرف

د. لؤي إدور جورج

شباط ٢٠٠٧ صفر ١٤٢٨