CHAPTER 3

APPROACH & METHODOLOGY

3.1 ALGORITHMIC APPROACH

The encryption of audio signals is a task that requires the utmost optimisation in the user and sender's end. Keeping this in mind we did a survey of many literature papers and articles and landed on the conclusion that in order to maximize the efficiency and minimize the system load on which the program will run it is very important that we find the best suitable algorithm to solve the problem. Therefore after reading about 20+ papers on the matter subject we handpicked 6 such papers that guided us towards a solution/approach we needed.

These papers were titled as mentioned in the following order:

A. An Effective Watermarking Method Based on Energy Averaging in Audio Signals
B. Digital Watermarks for Audio Signals
C. Efficient watermarking algorithm for digital audio/speech signal
D. Digital audio watermarking for QoS assessment of MP3 music signals
E. A blind quantum audio watermarking based on quantum discrete cosine transform
F. Development of an audio watermarking with decentralization of the watermarks

The first paper [22] proposes a spread time echo method by using pseudo noise sequencing for digital watermarking and uses the phenomenon of repetition of sound on reflection from an obstacle and encrypts it using PRN, also known as pseudo random noise. This method uses a signal on the same length as a noise and satisfies it using statistical randomness. Therefore since this algorithm uses statistical randomness, it requires the software to run many patterns and regularities that are non recognizable and therefore rely on the mean of the statistical value. We therefore came to the conclusion that this will use much of the system resources and is hence not variable for low end users.

The second paper [23] elaborates the term "pirate" and how PN sequencing can easily decrypt normal encryption and therefore why it is important to do digital - watermarking. Ignoring the motivation of the paper and focusing solely on the method it discusses we find that the method of focus is PN sequencing. PN sequencing stands for Pseudo-random Noise sequence and uses sets of bits that are statistically meant to be random. This Approach is an improvement on the previous approach as it depends on a function that can be utilized by any software and can run even on low end systems.

The third paper [24] discusses "a blind and audio/speech watermarking algorithm that combines the discrete Tchebichef moment transform (DTMT) of the mixed linear–nonlinear coupled map lattices (MLNCML), and discrete wavelet transform (DWT). In addition, the adopted strategy has a blind nature, where no original audio/speech is needed in watermark extraction."

This paper came from M. Yamini, H.Karmouni and M.Sayyouri and uses a very well optimized algorithm that not only improves the process of audio encryption but at the same time provides Tchebichef moment transform (DTMT) method to compress the digital images. Although this algorithm is extremely fast and provides a solution for the image compression as well, we voted to not use this as the scope of mathematics required to implement the method is way beyond our scope and will not meet/line up with the deadline of the project.

The fourth paper [27] discusses “Digital audio watermarking for QoS assessment of MP3 music signals.” This is another paper that discusses the type of audio signal that we are aiming at. And at the same time this paper proposes an watermarking signal technique to provide a quality assessment of the received audio signal after coding. But this paper has no mention of any encryption algorithm but uses signal processing to assess the quality of the signal. But since this was the only paper (among few) that talks about the quality assessment of audio signal using an open source software we found this approach to be quite helpful for using in the testing phase of our project.

The fifth paper discusses [31] A blind quantum audio watermarking method based on quantum discrete cosine transform. It uses the integration of LSB (least significant bit) and MSB (most significant bit) based on quantum discrete cosine transform (qDCT). The quantum discrete cosine transform expresses a finite sequence of data points in terms of a sum of cosine functions oscillating at different frequencies. This very advanced approach as it requires a very large dataset to fine tune the regression value. It also requires some capital to get the dataset and the software required to work on. But since our work is focused on an open source approach we decided to not go with this approach.

The sixth paper talks about the Development of an audio watermarking with decentralization of the watermarks. “The proposed technique is executed by incorporating multi-level DWT along with the use of multiple images with different sizes as watermarks.” This approach uses Discrete Wave Transform. In general sense a Discrete Wave Transform always returns only a single coefficient to its approximation value. Using the same on a multi level signal allows the extraction of multiple time-frequency features from a time series by decomposing the series as low and high frequency level - by - level. This is also a wonderful approach that does not over complicate the signal transposing value. However using discrete signal analysis will require additional hardware components (ADC - DAC Filters) to be integrated and this will in result require each sender and receiver encrypting their audio to buy/install a hardware component. Therefore it would increase the cost of encryption which will limit the benefit to only a handful.

Therefore after analyzing the approach discussed in the above mentioned 6 papers we have come up with the decision to use the method discussed in the 4th paper (QoS assessment) and use the quality assessment of signal processing and how to improve on it from the second paper (PN sequencing). We found that using PN sequencing to test the encryption with digital watermarking proves to be the most fruitful as pseudo-noise code (PN code) or pseudo-random-noise code (PRN code) has a spectrum similar to the random sequence of bits that the random function provides.

3.2 ABIDING SERVICES

Before using the QoS assessment encryption it is important to understand the QoS service. The QoS is known as the quality of services and refers to any technology that deals with data transmission. It is in place to reduce the packet loss, jitters and latency on the network. There are priorities in place that every transmission must follow to abide by these protocols.

QoS is used by organizations to meet the requirements of traffic sensitive applications which use real-time voice and video, and may as well be used to minimize quality deprecation caused by as mentioned before: packet loss, delay, and jitter

Therefore while encrypting the audio it is absolutely necessary that the class of service is strictly followed (CoS). This comprises of certain quantitative parameters such as:

a. **Packet loss**
b. **Jitters**
c. **Latency**
d. **Bandwidth**
e. **Mean Opinion score** (Also known as MOS. This rates the voice quality using a 5 point scale. Where 5 is the highest value and therefore represents the highest quality.)

Table 3.1 - QoS Parameters Testing

PARAMETER	EXPECTED	ACTUAL	TESTED ON
ISP Packet Loss	● 212 / 228 Bytes ● 5 Seconds	144 bytes (77% of expected), 10 seconds (282% better)	https://packetlosstest.com
Latency	Range = 80ms - 100ms	87ms	https://superpowered.com/weblatency
Delay	Acceptable Delay: 24 Milliseconds	20 milliseconds	https://packetlosstest.com
Distortion	Acceptable Delay: 7%	6%	*(No test available (needs manual monitoring)*

3.3 ADVANTAGES

The essential advantage of QoS is that it guarantees the availability of servers and the services that depend on it. It facilitates the secure and convenient movement of data over the network.

a. Mission-critical applications have access to the resources they require.
b. Administrators can manage traffic better.
c. Organizations can reduce costs by eliminating the need to purchase new network infrastructure.
d. User experience is improved.

Performance of service tools select packets in order to make the most of their network's limited bandwidth. In other words, the connectivity can only transfer so much data in a given length of time. As a result, QoS tools select packets in such a way that bandwidth is managed to give the best network infrastructure feasible in the time allotted.

To classify packets, a QoS spatial analysis analyzes the packet headers chunks of data that tell the application and its communication ports whatever the register holds and what it is being transmitted for. However, for our project which uses digital watermarking of audio signals, we will not require the IP address of the receiver to be used in the code as that function is provided by the transmission services and they take care of such stuff due to the compliance of their routing protocols.

The QoS can also scan incoming packets and will prompt if a packet is relevant while choosing it above packets that are less time-sensitive. The shipping and return ids on something like a physical package can be referred to as packet headers.

There are two types of digital watermarking- visible and invisible. The visible watermark is similar to the corporation logo displayed at its letterhead but the invisible one that is embedded in the media is unnoticeable and undetectable to the human eye. There are some six types of watermarks- visible, non-visible, private, public, perceptual and bit stream. This is a blind watermarking technique that meets the requirements of invisibility and robustness.

3.4 IMPLEMENTATION

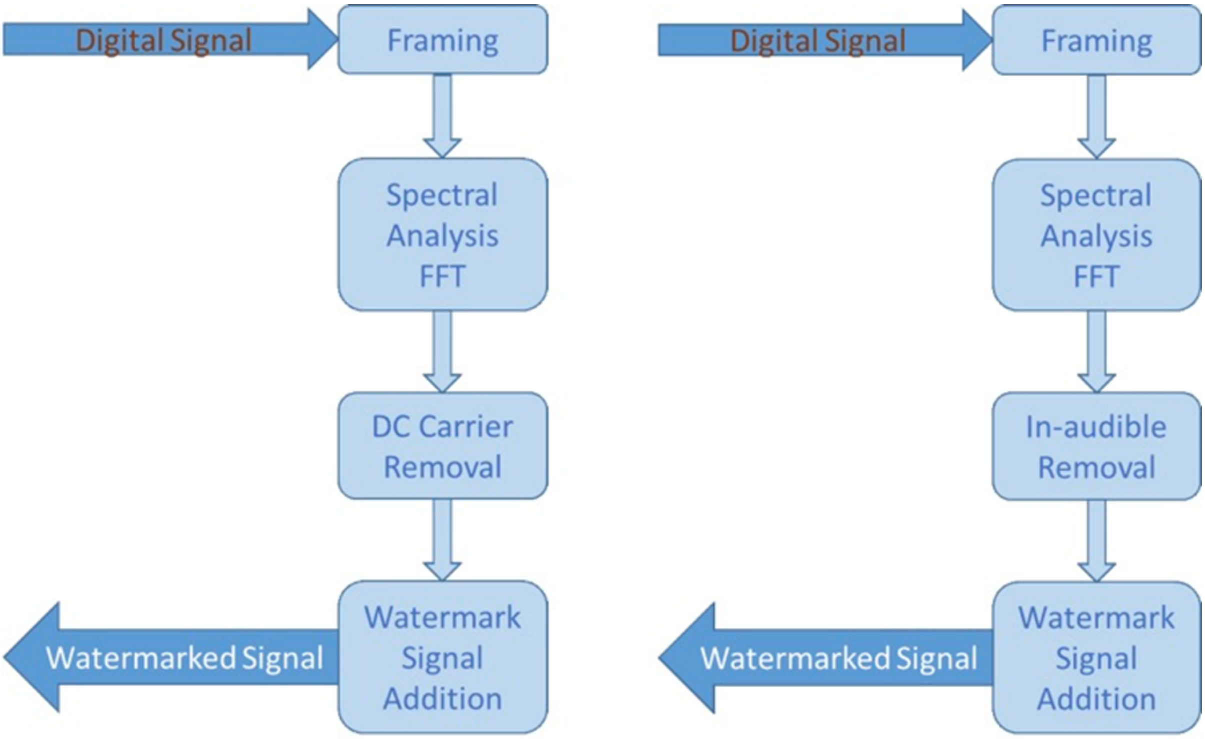


Figure 3.1 - Algorithmic Implementation

A Watermark is embedded into the digital signal at each point of distribution which is unnoticeable to the human eye. If the data is copied the watermark is carried with the data and the watermark can be retrieved from the copy and the source of distribution is known. Digital Watermarking is a technology that embeds machine readable information within the content of a digital media file that could be image, audio or video. The information is encoded through subtle changes to the image, audio, or video.

In other words the watermarking is a practice of modifying the digital data(software program, photos, songs, videos) without causing destruction to it to embed a message about that work. Multimedia watermarking is the practice of imperceptibly altering a work. The watermarking is a technique related to steganography which means keeping the existence of messages secret by hiding them within objects, media, or other messages.

3.5 TOOLS AND MECHANISM

QoS mechanisms are classified according to the functions they perform in network management.

I. **Classification and marking** - Helps to distinguish and arrange packets into distinct traffic kinds. It identifies every transaction as either component of a network class, allowing networks to identify the packet's class. These are executed on devices like routers, switches, and access points.

II. **Congestion management** - To identify which priority to place packets in, these technologies employ packet categorization and marking.

III. **Congestion avoidance** - When there is congestion in the network, these programmes monitor it and discard low-priority packets. Evenly distributed irregular rapid identification as well as random early intervention are two strategies for avoiding traffic congestion.

IV. **Shaping** - These technologies alter network traffic and favor less time-sensitive apps like email, messaging. Buffers, Generic Traffic Shaping, and Frame-Relay Traffic Shaping are examples of traffic shaping technologies.

V. **Link efficiency** - These programmes optimize bandwidth use and minimize network packet latency. Link efficiency methods, while not just for QoS, are used in combination with other QoS strategies. Authentic Routing Algorithm, header decompression, Tcp, header compression, and link compression are examples of link efficiency technologies.

QoS tools has 6 categories:

I. **Classification** - Identifies transmitted traffic and marks it to see that other transmissions can identify it.

II. **Queueing** - Keeps store of packet for later use (our project however will not require queuing).

III. **Policing** - Avoids congestion as on the specific bandwidth. This is of utmost importance as it makes the watermark less clogged.

IV. **Shaping** - Unlike queuing it processes the xcess traffic and does not completely discard it.

V. **WRED** - Stands for Weighted random early discard and prioritizes high priority data from the already negative network congestion

VI. **Fragmentation and compression** - Prevents the delay by minimizing the bandwidth.

CHAPTER 4

WORKING & ANALYSIS

4.1 PLANNING

For the working of our project we came up with a methodology framework and divided it into 3 months worth of work. An “.io” diagram showing our framework is attached below:

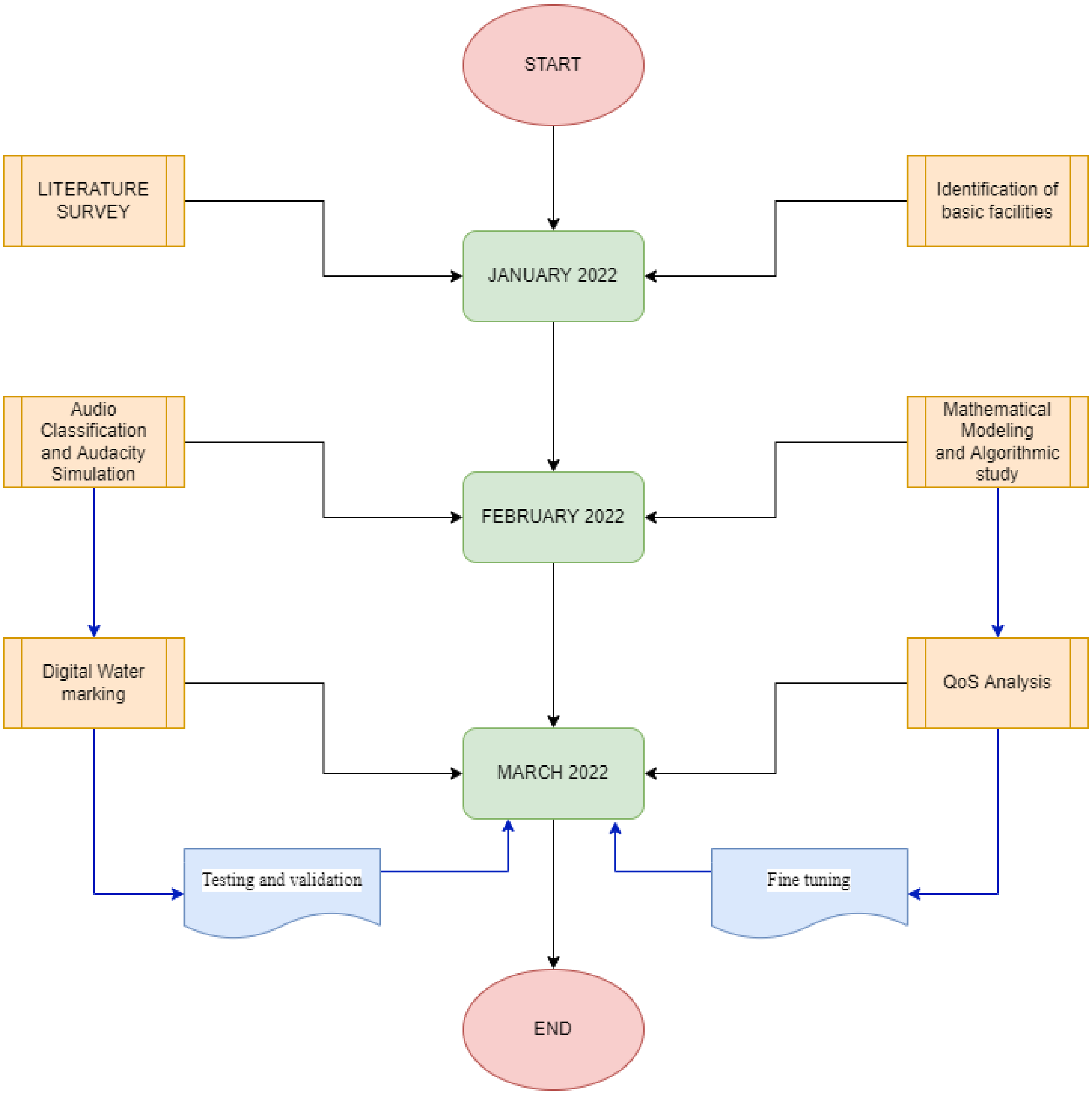


Figure 4.1 - Working Flowchart

We started with the classification of the audio signal and dividing it into things such as bandwidth, nominal level, power level in decibels (dB), and voltage level. The attachment between P-V is the impedance of the signal path. Signal paths may be 1 ended or balanced.

For our work we will be using MATLAB and Audacity which is an open source - multi platform software. This is an inhouse project for which the minimum specifications required to run the project are as follows:

Table 4.1 - Parameter Specifications

Parameter	**Minimum Specifications Needed**
Operating System (OS)	Windows 7
Processing power	1.8 GHz
Random Access Memory (RAM)	2 GB

We started with using Audacity to test the form of audio signal and the Waveform Audio File Format in the preliminary stages. We used a file that contained audio recordings with different sampling rates and bit rates but were saved in a 44.1 kHz, 16-bit format on Audacity. These files were divided into host and port WAV's and sampled onto Audacity and was mapped using the microsoft sound mapper at the following specifications:

Table 4.2 - Parameter Value Set

Parameter Name	**Value Set**
Audio Channel	Mono
Frequency	44100 Hz
Bit point (floating)	32 bit
Mapping (range)	- 1.0 - 1.0
Time Sampled	Duration = 60 seconds

During this we found that any traditional form of watermarking on an audio file can be easily decrypted at 'line level' using software. To prove this we added a water mark at one junction and decrypted the signal using the "same software" and varying the parameters mentioned above.

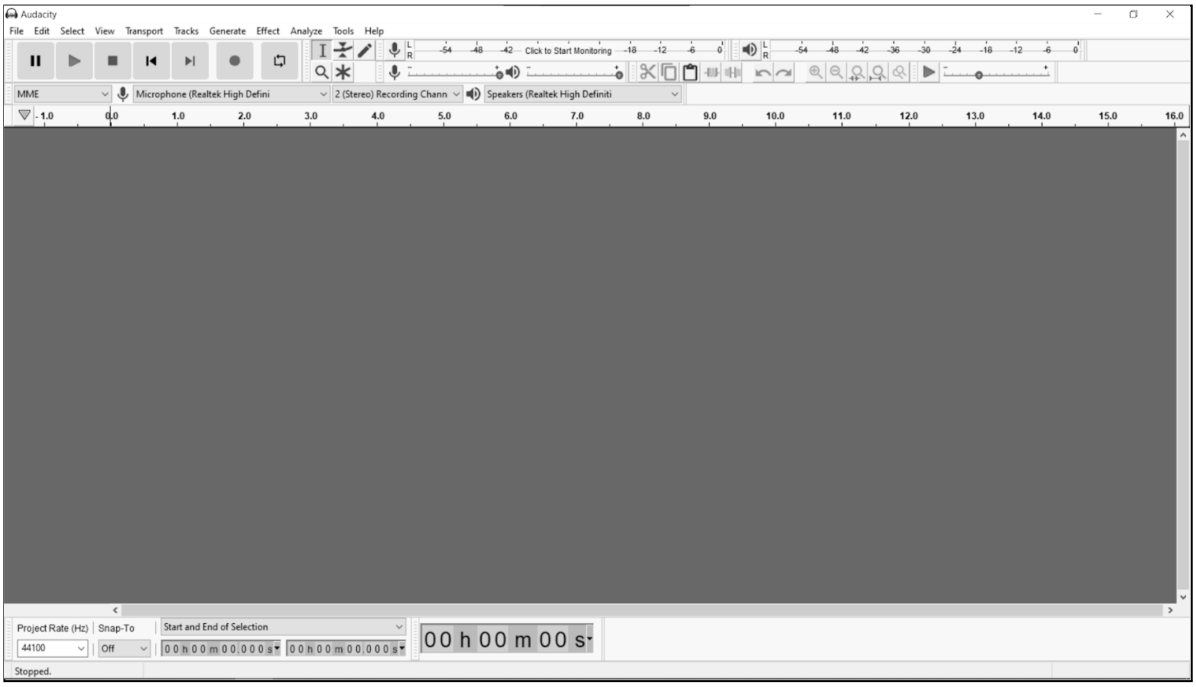
And found that the watermark (called as sis - gen) could easily be retrieved. Then we varied it with 4 other forms of waves (sine, square, sawtooth, triangle) and took the readings (comparison is yet to be done) with gain set at +5 dB and frequency at 440 Hz.

4.2 ENCRYPTION METHODS

4.2.1 USING OVERLAPPING

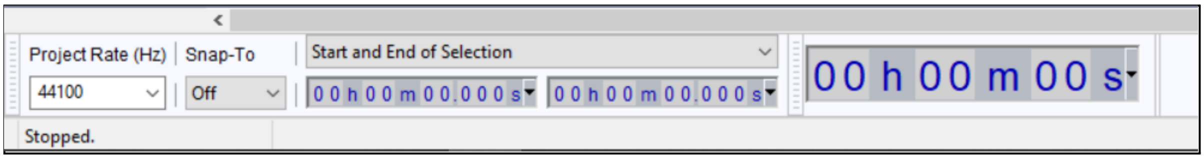
During our literature survey we came across an existing process of signal encryption that uses a software or tool to overlap an audio signal with another audio signal before transmission and then proceeds to remove that same overlapped signal on the receiver's end.

We used a similar software called Audacity to replicate the same and to find out for ourselves whether the theoretical flaws we found were practical or not.


Audacity
File Edit Select View Transport Tracks Generate Effect Analyze Tools Help
Click to Start Monitoring
MME
Microphone (Realtek High Defini
2 (Stereo) Recording Chann
Speakers (Realtek High Definiti
-1.0 0.0 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0 15.0 16.0

Project Rate (Hz): 44100
Snap-To: Off
Start and End of Selection: 00 h 00 m 00.000 s – 00 h 00 m 00.000 s
00 h 00 m 00 s
Stopped.

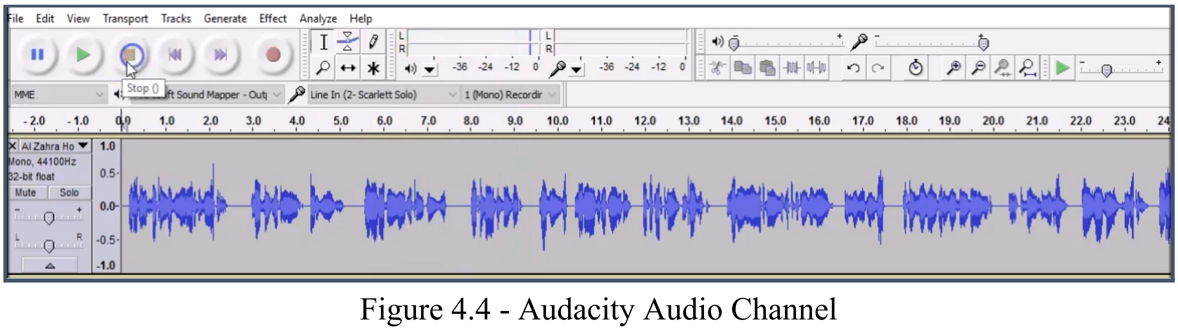
Figure 4.2 - Audacity Homepage



Project Rate (Hz): 44100
Snap-To: Off
Start and End of Selection: 00 h 00 m 00.000 s – 00 h 00 m 00.000 s
00 h 00 m 00 s
Stopped.

Figure 4.3 - Audacity Editables

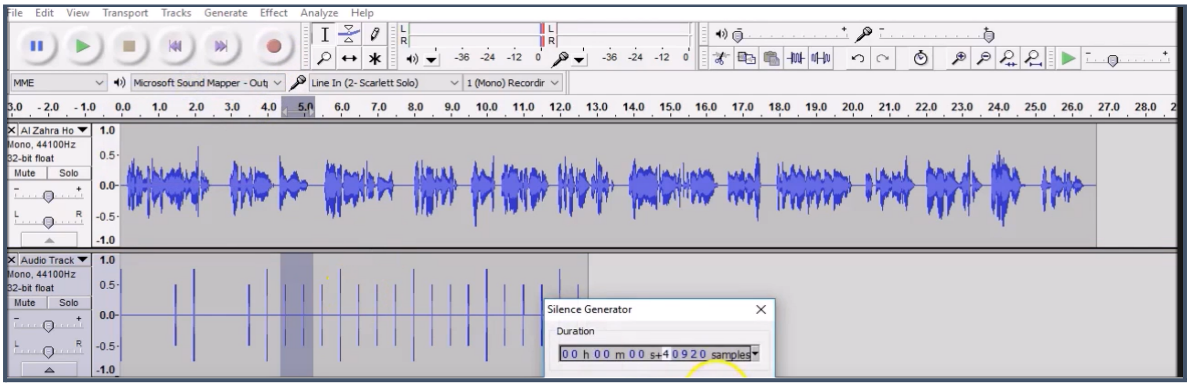
For our audio we used a 1 minute clip from the audiobook called “The Adventures Of Sherlock Holmes”. This is an open source audio book available for sampling and free using. We added that audio file as a mono stereo file as tract one. Then we generated a secondary white noise and added it into track two (just below the previous track).



Al Zahra Ho
Mono, 44100Hz
32-bit float

Figure 4.4 - Audacity Audio Channel

Then we generate a tone (from the software) and superimpose it on the original audio file.In audacity we can basically change every single parameter of an audio file. For our sampling test we changed it to the following settings: The result screenshot is attached below:


File Edit View Transport Tracks Generate Effect Analyze Help
MME
Microsoft Sound Mapper - Out
Line In (2- Scarlett Solo)
1 (Mono) Recordir
Al Zahra Ho
Mono, 44100Hz
32-bit float
Mute Solo
Audio Track
Mono, 44100Hz
32-bit float
Mute Solo

Silence Generator

Duration

00 h 00 m 00 s+40920 samples

Figure 4.4.2 - Overlapped audio

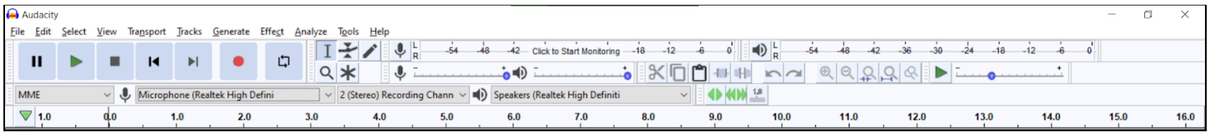

Audacity
File Edit Select View Transport Tracks Generate Effect Analyze Tools Help
Click to Start Monitoring
MME
Microphone (Realtek High Defini
2 (Stereo) Recording Chann
Speakers (Realtek High Definiti

Figure 4.5 - Audacity toolbar

We then generated the traditionally encrypted audio signal and studied its pros and cons. We believe that encrypting an audio file like this has several advantages and disadvantages.But in the domain of encrypting an audio signal the disadvantages top the advantages. Some of the disadvantages are which are:

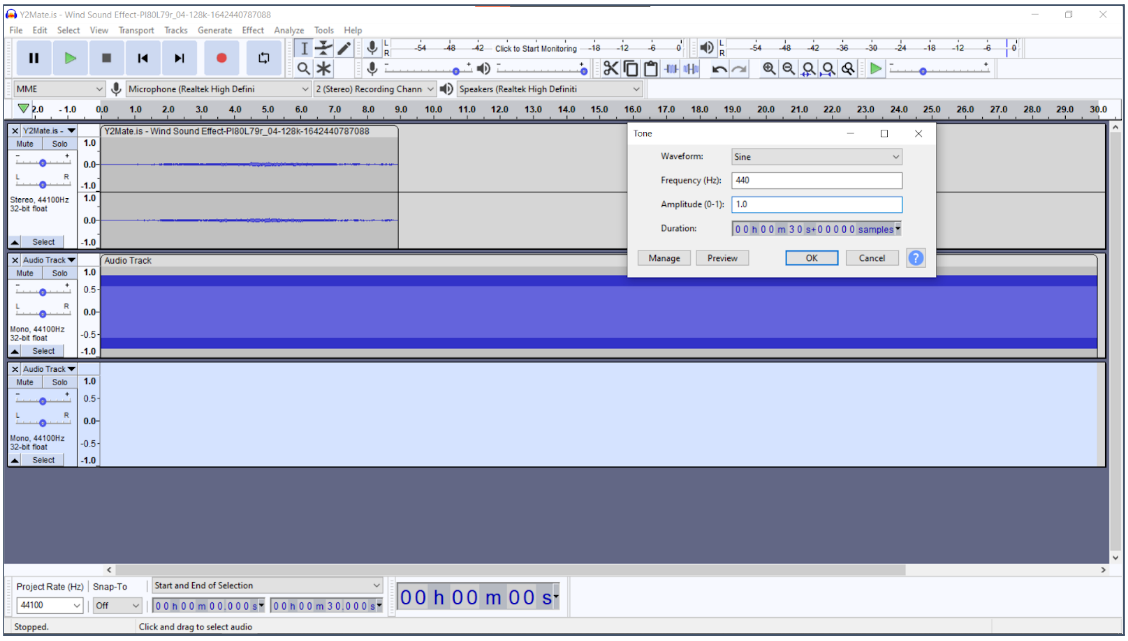
1. The original audio signal will be susceptible to noise.
2. Anyone with the audio file can easily see the tune used to encrypt the file and remove it using a mediocre level software.
3. The sender has almost no control over the encryption.
4. The raw audio signal gets quite messy to discern the original content of the file

Therefore with these crucial disadvantages we proceeded to test it with another method to see if this method actually be improved upon and whether digital watermarking is actually needed.

4.2.2 USING WAVEFORM VARIATION

From the first method we saw that one of the major disadvantages was that “A TONE” was being used as the overlapping signal. This in turn generated noise that made the original audio quality bad. However what if we used a single waveform rather than a whole tone. We used Audacity as before to do the same. We kept our original audio the same but changed the overlapping signal from a tone to a single waveform. We did this with 4 separated waveforms (sine, square, sawtooth and triangle).

Firstly we used a sine wave (400Hz, 1.0 Amplitude). From this we find that even though the raw audio file ended up retaining its original audio quality it showed no improvement in terms of encrypting the audio. All the open loopholes that were present when using tone overlapping are still present when using a waveform overlapping. The same result was seen while using square waveform, sawtooth waveform, triangle.


Tone

Waveform: Sine

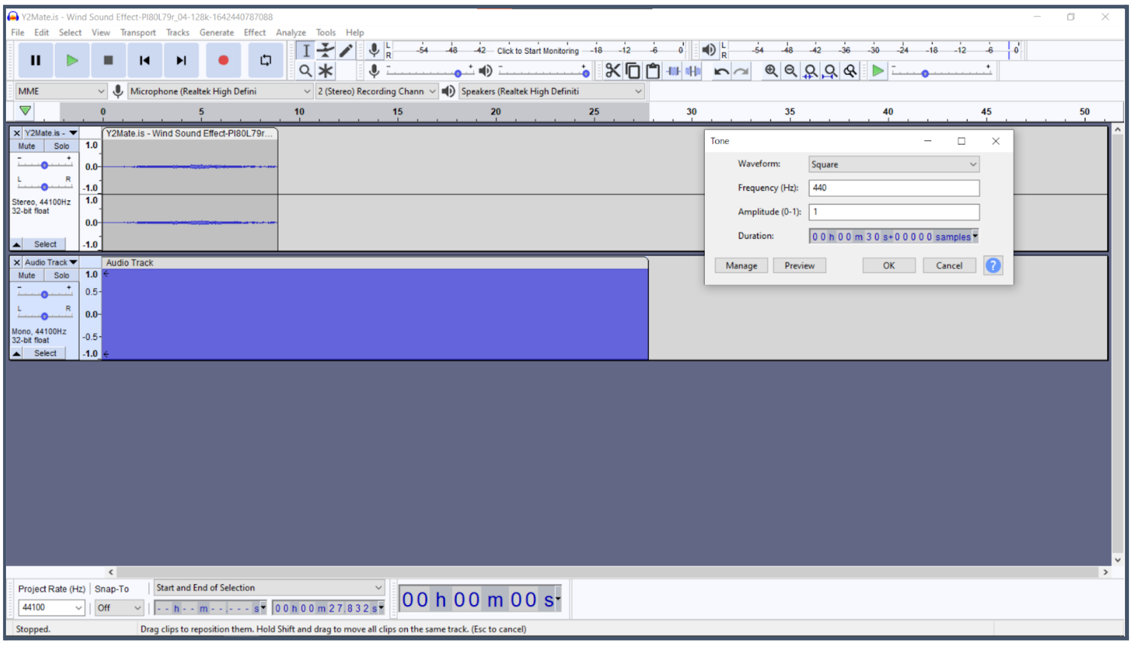
Frequency (Hz): 440

Amplitude (0-1): 1.0

Duration: 00 h 00 m 30 s+00000 samples

Y2Mate.is - Wind Sound Effect-PI80L79r_04-128k-1642440787088
File Edit Select View Transport Tracks Generate Effect Analyze Tools Help
Click to Start Monitoring
MME
Microphone (Realtek High Defini
2 (Stereo) Recording Chann
Speakers (Realtek High Definiti
Audio Track
Mute Solo
Stereo, 44100Hz 32-bit float
Mono, 44100Hz 32-bit float
Select
Manage Preview OK Cancel
Project Rate (Hz) 44100
Snap-To Off
Start and End of Selection
00 h 00 m 00.000 s
00 h 00 m 30.000 s
00 h 00 m 00 s
Stopped.
Click and drag to select audio

Figure 4.6 - Sine wave overlapping


Tone

Waveform: Square

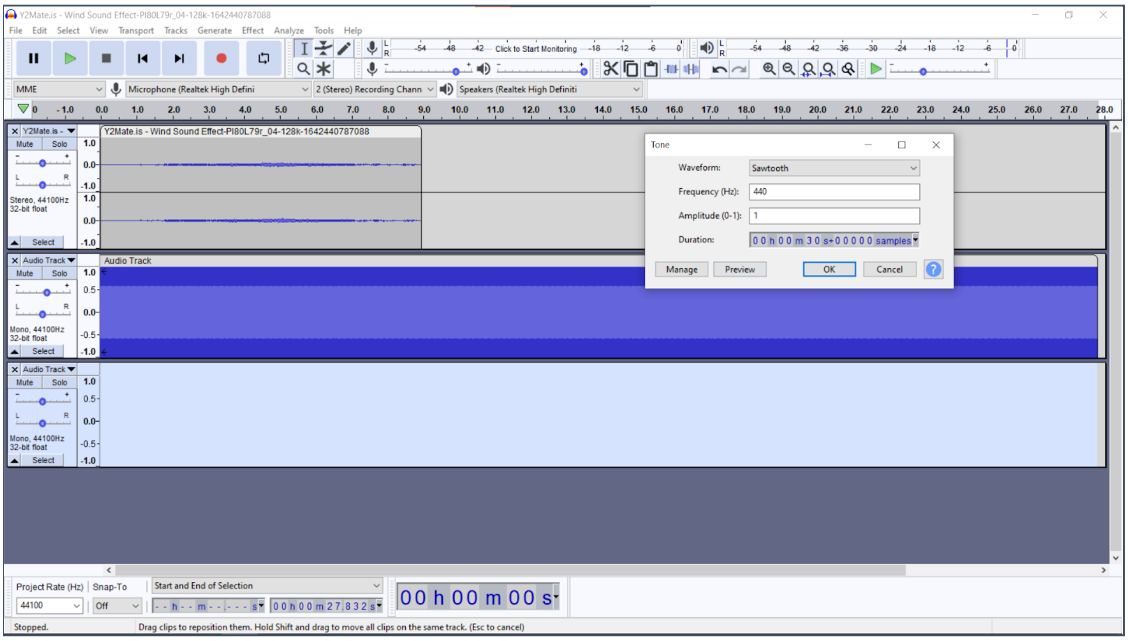
Frequency (Hz): 440

Amplitude (0-1): 1

Duration: 00 h 00 m 30 s+00000 samples

Y2Mate.is - Wind Sound Effect-PI80L79r_04-128k-1642440787088
File Edit Select View Transport Tracks Generate Effect Analyze Tools Help
Click to Start Monitoring
MME
Microphone (Realtek High Defini
2 (Stereo) Recording Chann
Speakers (Realtek High Definiti
Audio Track
Mute Solo
Stereo, 44100Hz 32-bit float
Mono, 44100Hz 32-bit float
Select
Manage Preview OK Cancel
Project Rate (Hz) 44100
Snap-To Off
Start and End of Selection
00 h 00 m 27.832 s
00 h 00 m 00 s
Stopped.
Drag clips to reposition them. Hold Shift and drag to move all clips on the same track. (Esc to cancel)

Figure 4.7 - Square wave overlapping


Y2Mate.is - Wind Sound Effect-PI80L79r_04-128k-1642440787088

Tone

Waveform: Sawtooth

Frequency (Hz): 440

Amplitude (0-1): 1

Duration: 00 h 00 m 30 s+00000 samples

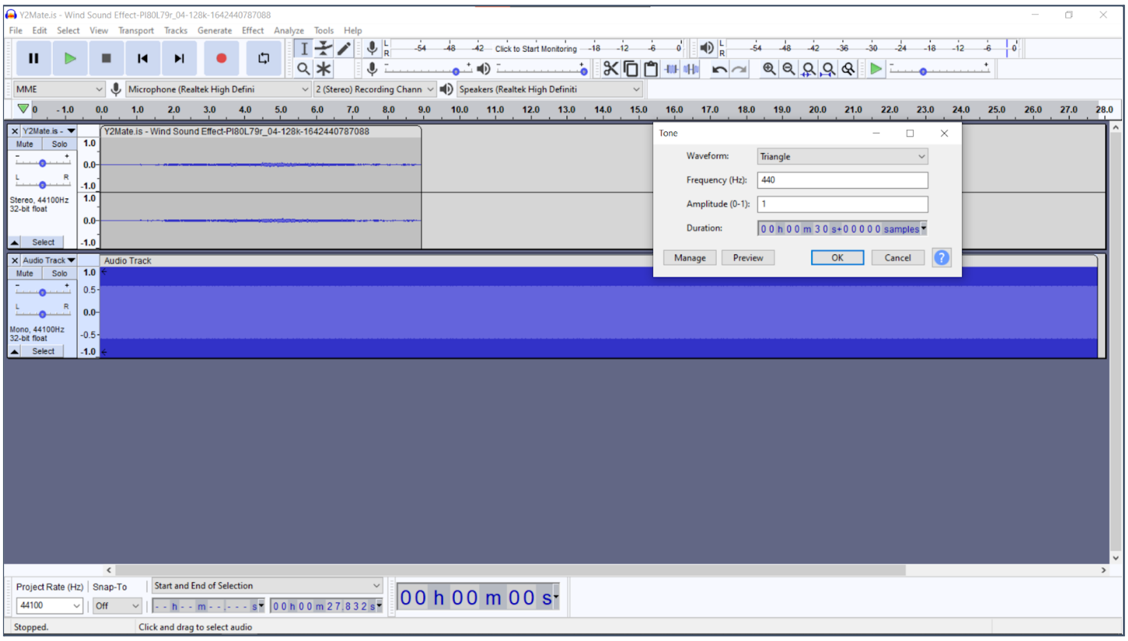
Project Rate (Hz): 44100

Snap-To: Off

Stopped.

Drag clips to reposition them. Hold Shift and drag to move all clips on the same track. (Esc to cancel)

Figure 4.8 - Sawtooth wave overlapping


Y2Mate.is - Wind Sound Effect-PI80L79r_04-128k-1642440787088

Tone

Waveform: Triangle

Frequency (Hz): 440

Amplitude (0-1): 1

Duration: 00 h 00 m 30 s+00000 samples

Project Rate (Hz): 44100

Snap-To: Off

Stopped.

Click and drag to select audio

Figure 4.9 - Triangle wave overlapping

Therefore after testing it with waveforms we are still left with 3 major disadvantages to this type of audio encryption that make it susceptible to 3rd party attacks and decryption:

1. Anyone with the audio file can easily see the tune used to encrypt the file and remove it using a mediocre level software.
2. The sender has almost no control over the encryption.
3. The raw audio signal gets quite messy to discern the original content of the file

4.2.3 USING LINEAR REGRESSION

Finally we use a testing method of regression based method to visualize the distribution of the model:

Plot Data

1. Ddata points X and y into a new figure.
2. It will plot the data points with + for the positive values and o for the negative values. X is assumed to be a Mx2 matrix.
3. Plotting the '+' AND '-' values on a 2 DIMENSION plot, using the option 'k+' for the '+' values and 'k(0)' for the '-' values.

Regression Watermarking

1. Segmentation Plotting
2. Logistic Regression
   a. Setup the data matrix
   b. Add intercept term to x and X_test
   c. Initialize fitting parameters
   d. Compute and display initial and final values of the gradient
   e. Compute and display gradients with != 0 theta value.
3. Optimizing using fminunc
   a. Set options for fminunc
   b. Click fminunc to obtain the optimal value for the theta
   c. Print theta to screen
   d. Plot Boundary
   e. Put some labels
   f. Specified in plot order
4. Training Set
   a. Accuracy on training set

Plot Decision Boundary

- Plots the data points X and y into a new figure plot with the decision boundary defined by the theta value.
- Plots the data points with + for the '+' values and 0 for the '-' values. Mx3 matrix, where the first column is an all-ones column for the intercept.
   a. MxN, N>3 matrix, where the first column is all-ones.
- Plot Data
   a. Only need 2 definite points to define a line
   b. Calculate the decision boundary line
   c. Plot, and adjust axes for better viewing
   d. Legend, specific for the distribution
   e. Grid range

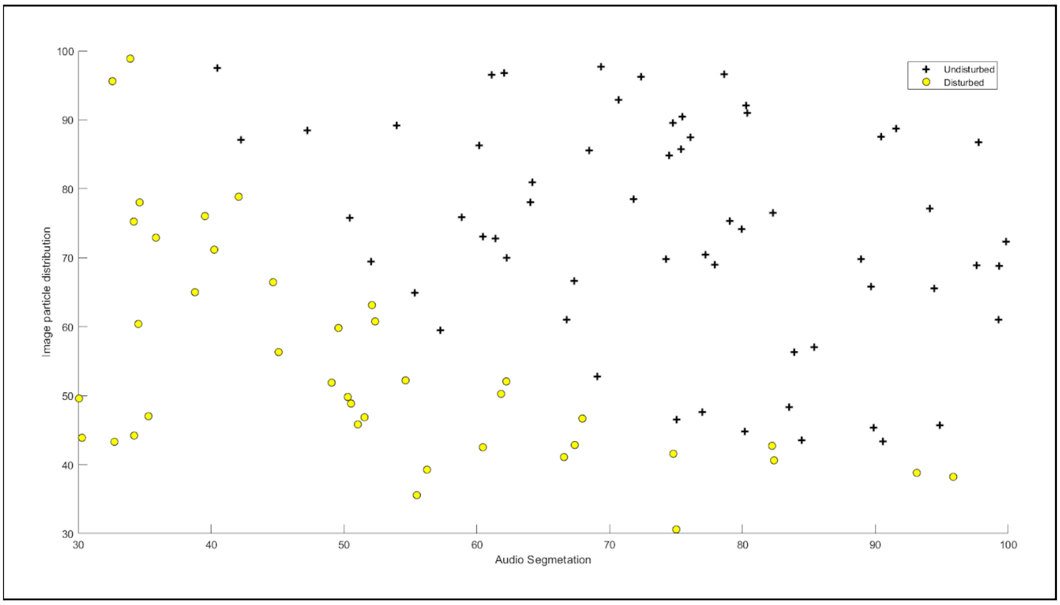


Figure 4.9 - Audio Segmentation Stage 1

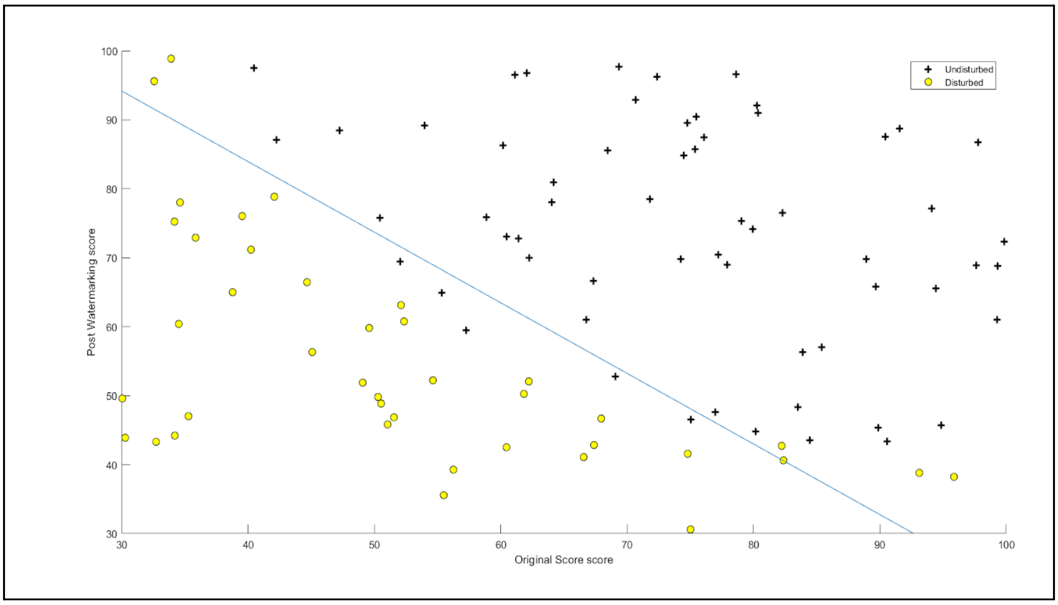


Figure 4.9.1 - Audio Segmentation Stage 1

But we see that even this method reached something called a Local Minimum possible. Because of this, we have reached a local minimum, but cannot be certain because the '1O' optimality measure is not less than the Optimality Tolerance tolerance.

Also This method becomes extremely tedious when the data set is large. This adds to our claim that using digital watermarking for audio encryption is the best path possible.

Table 4.3 - Regression based results

Difference in initial theta value	+ 0.693147
Expected difference:	+ 0.69
Gradient at initial theta value for its zeros	- 0.100000 - 12.009217 - 11.262842
Expected gradients:	- 0.1000 - 12.0092 - 11.2628
Value at theta taken at test: Expected :	+ 0.218330 + 0.218
Gradient theta taken at test:	+ 0.042903 + 2.566234 + 2.646797
Expected gradients:	+ 0.043 + 2.566 + 2.647

4.3 CODING

Therefore after analyzing the previous outputs and studying their disadvantages we see that using digital watermarking of audio signals for enhanced signal protection is indeed one of the most convenient ways to go about the problem statement of audio encryption.

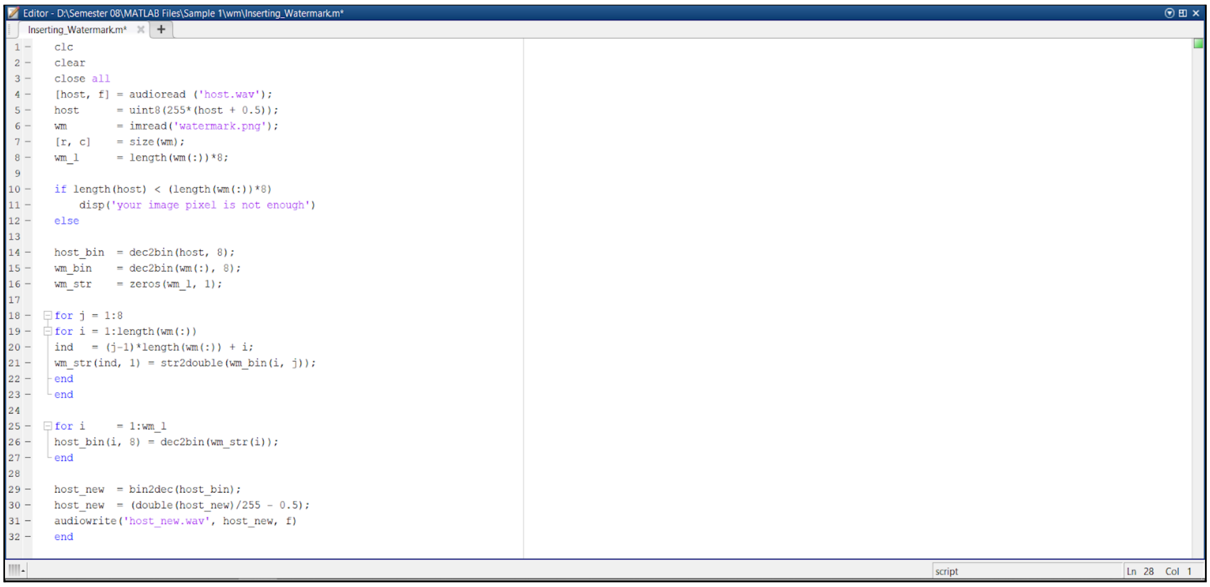
Hence for our project we have used MATLAB to achieve the same. The entire process is divided into parts of matlab code namely “Embedded the watermark” and “Extracting the water marking”. It was important to use two separate code scripts for this process as mentioned in the QoS literature review, it protects the integrity of the code snippet in case some “pirate” affects one section of the code, the others will remain intact and free of interference from the 3rd party.

Our project includes 2 MATLAB function files (write) and 1 sample MP3 files (read). It works like the commands WAVWRITE and WAVREAD. It must be noted that this version was made in MATLAB for WINDOWS only. Also note that we are using DOUBLE data type with double [-0.5 +0.5] to 'uint8' [0 255] as the parameter and this can be changed according to the receiver's need or the size of the watermark./host signal.

4.3.1 INSERTING WATERMARK

The insert watermark code script follows the following hierarchical approach:

1. Clear memory and command window
2. Load data
3. Watermarking
4. Prepare host
5. Prepare watermark
6. Insert watermark into host
7. Watermarked host
8. Save the watermarked host



Editor - D:\Semester 08\MATLAB Files\Sample 1\wm\Inserting_Watermark.m*
Inserting_Watermark.m*
script
Ln 28 Col 1

```
  clc
  clear
  close all
  [host, f] = audioread ('host.wav');
  host      = uint8(255*(host + 0.5));
  wm        = imread('watermark.png');
  [r, c]    = size(wm);
  wm_l      = length(wm(:))*8;

  if length(host) < (length(wm(:))*8)
      disp('your image pixel is not enough')
  else

  host_bin  = dec2bin(host, 8);
  wm_bin    = dec2bin(wm(:), 8);
  wm_str    = zeros(wm_l, 1);

for j = 1:8
for i = 1:length(wm(:))
  ind   = (j-1)*length(wm(:)) + i;
  wm_str(ind, 1) = str2double(wm_bin(i, j));
  end
  end

for i     = 1:wm_l
  host_bin(i, 8) = dec2bin(wm_str(i));
  end

  host_new  = bin2dec(host_bin);
  host_new  = (double(host_new)/255 - 0.5);
  audiowrite('host_new.wav', host_new, f)
  end
```

Figure 4.9.2 - Inserting watermark code

Working of the code:

1. First of all we run a command to clear all the previous system clutter(memory) and get a clean command window.
2. Then we use the “audioread” command to load the host audio and store it in a variable with its address stored into an index variable (which in our code is named as f).
3. Then we load the watermark image (which is a png) into a using the “imread” command and store it is a variable named “wm”
4. Then we use a 2D array to store the dimensions of the image we are using as a watermark.
5. Then we use a conditional statement to check if the length of the host file is less than the length of the watermark * 8.
6. If it is then we reject that image as a watermark and display a message saying to use a different watermark.

7. Then we call the predefined MATLAB function called “dec2bin” to start the binary host.
8. Then we write the code to prepare the watermark:
    a. Firstly we call the dec2bin function used just above this function.
    b. Then we declare a n X 8 zeroes matrix (where n in the proportional row size

of the watermark).

c. Then we run a for loop to insert a watermark into the first plane of the host signal.

d. Then for every iteration of the for loop declare the LSB by using the command “host_bin(i, 8) = dec2bin(wm_str(i))”

9. Finally host the watermark using the bin2dec() function and specify the data type as “double”.

10. End the code block.

11. Save the host code.

12. RUN the code to insert the watermark.

After running the code the workspace values obtained are attached below.

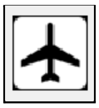
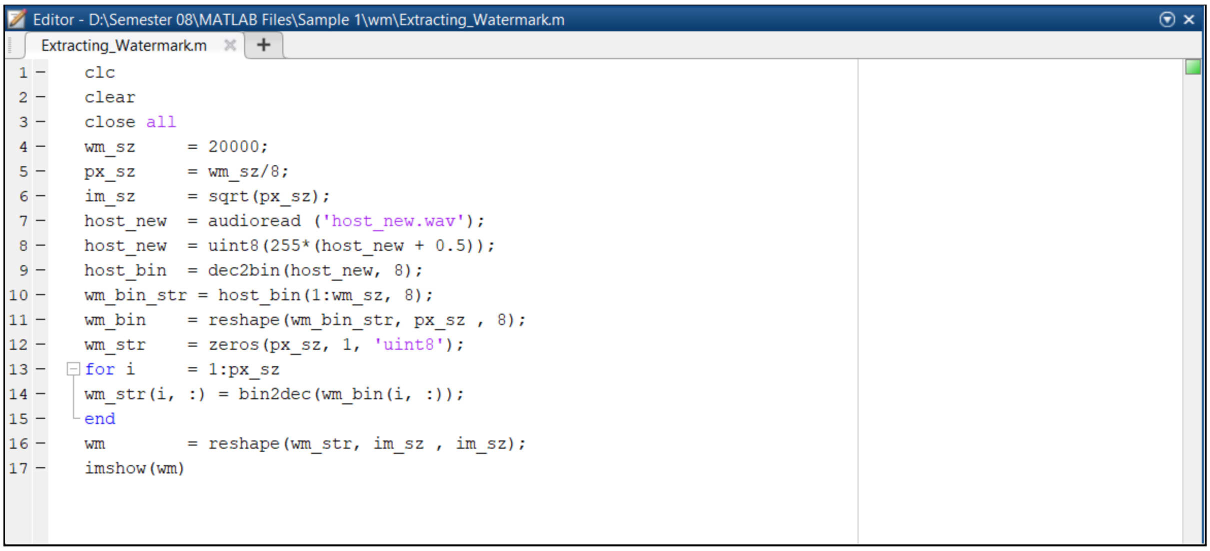


Figure 4.9.3 - Watermark

4.3.2 EXTRACTING WATERMARK

The insert watermark code script follows the following hierarchical approach:

- ➢ Clear memory and command window
- ➢ Load data
- ➢ Prepare host
- ➢ Extract watermark
- ➢ Display image and supporting html file

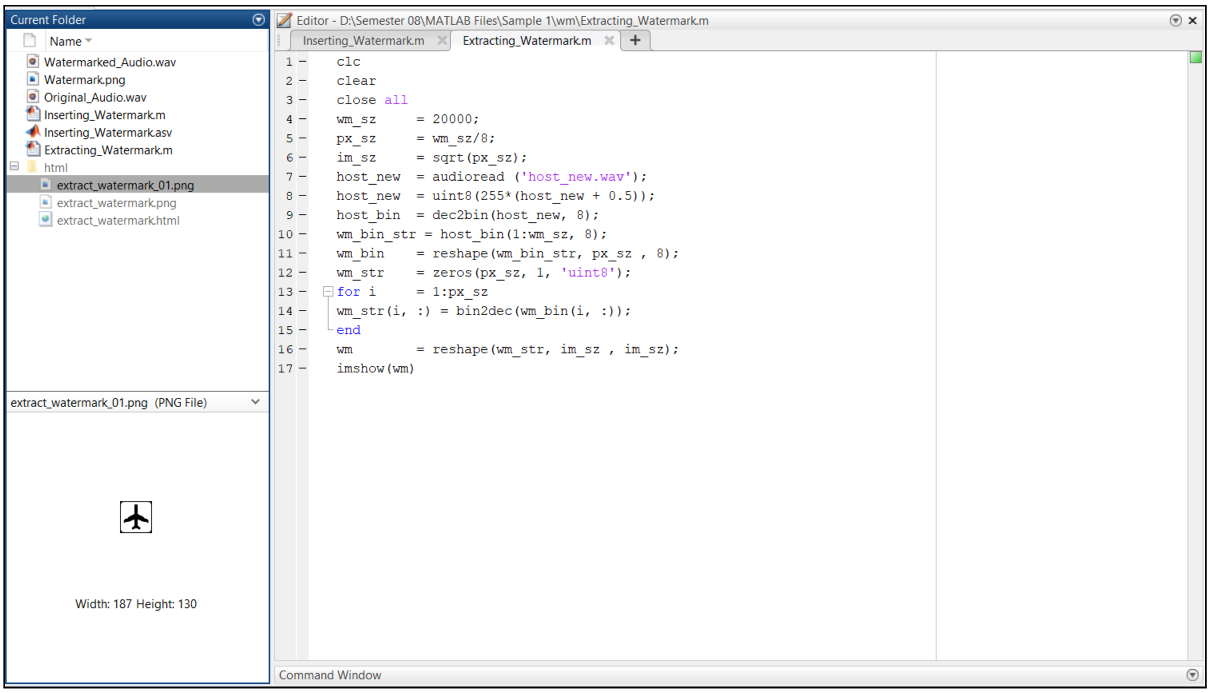

Editor - D:\Semester 08\MATLAB Files\Sample 1\wm\Extracting_Watermark.m
Extracting_Watermark.m

```
clc
clear
close all
wm_sz      = 20000;
px_sz      = wm_sz/8;
im_sz      = sqrt(px_sz);
host_new   = audioread ('host_new.wav');
host_new   = uint8(255*(host_new + 0.5));
host_bin   = dec2bin(host_new, 8);
wm_bin_str = host_bin(1:wm_sz, 8);
wm_bin     = reshape(wm_bin_str, px_sz , 8);
wm_str     = zeros(px_sz, 1, 'uint8');
for i      = 1:px_sz
wm_str(i, :) = bin2dec(wm_bin(i, :));
end
wm         = reshape(wm_str, im_sz , im_sz);
imshow(wm)
```

Figure 4.9.4 - Extracting watermark

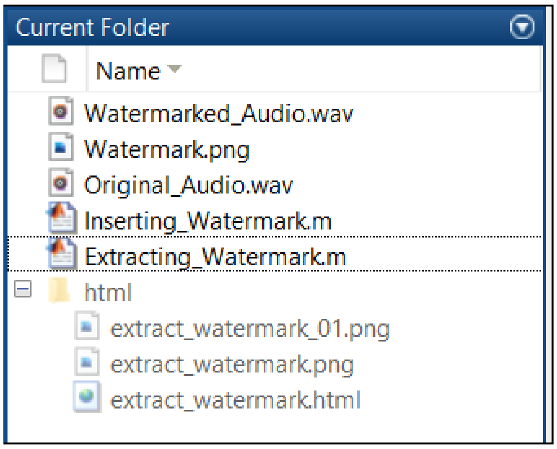
Working of the code:

1. First of all we run a command to clear all the previous system clutter(memory) and get a clean command window.
2. Specify an upper range for the watermark size and store it in a variable.
3. Divide that with 8 to get the number of pixels used in generating the size.
4. Store the image size in a variable as the square root of the pixel size.
5. Call the “audioread” function and pass the path of the watermarked host signal’s path name into the function.
6. Use the double [-0.5 +0.5] to 'uint8' [0 255] parameter to match the watermark dimensions matched to the double data type.
7. Define the host bin size at (1:max size) and store it in a variable.
8. Call the “reshape function” and pass the above variable along with the pixel size and the constant number 8(that we used to divide in step 2 into the function.
9. Define a zeros matrix.
10. Run a for loop on this matrix:
    a. For every iteration of the for loop use the bin2dec to generate a LSB and store it in a variable.
11. End the for loop.
12. Call the reshape function used before and replace the constant 8 with the image size variable.
13. Call in the imshow() function to finally extract and display the watermark.



```
clc
clear
close all
wm_sz      = 20000;
px_sz      = wm_sz/8;
im_sz      = sqrt(px_sz);
host_new   = audioread ('host_new.wav');
host_new   = uint8(255*(host_new + 0.5));
host_bin   = dec2bin(host_new, 8);
wm_bin_str = host_bin(1:wm_sz, 8);
wm_bin     = reshape(wm_bin_str, px_sz , 8);
wm_str     = zeros(px_sz, 1, 'uint8');
for i      = 1:px_sz
wm_str(i, :) = bin2dec(wm_bin(i, :));
end
wm         = reshape(wm_str, im_sz , im_sz);
imshow(wm)
```

Figure 4.9.5 Extracted watermark



Current Folder

Name

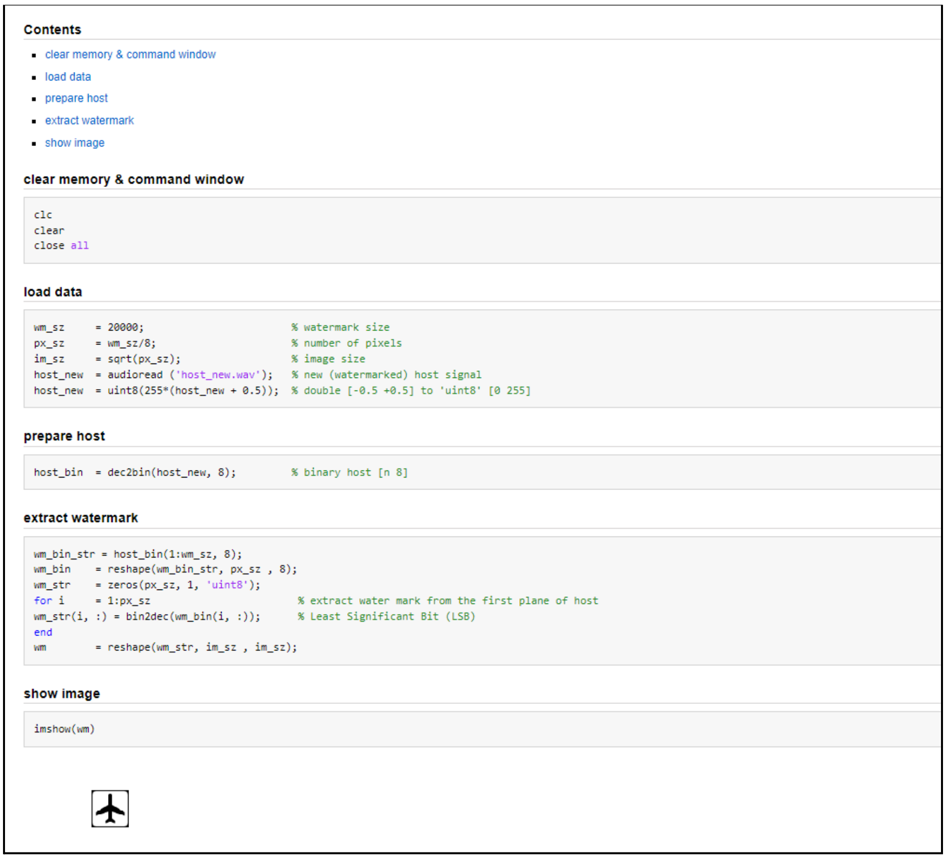
- Watermarked_Audio.wav
- Watermark.png
- Original_Audio.wav
- Inserting_Watermark.m
- Extracting_Watermark.m
- html
  - extract_watermark_01.png
  - extract_watermark.png
  - extract_watermark.html

Figure 4.9.6 - Folder screenshot

4.4 ANALYSIS

After the completion of the project we inferred that using digital watermarking for enhanced signal protection not only keeps the original quality of the sound unaltered but also encrypts the signal. On top of that the watermark stays hidden therefore it preserves the integrity of the encryption, making this approach a very fruitful form of audio encryption.

For starters not anyone who gets the encrypted audio can decode the audio as the sender has the numbers required to decrypt and extract the watermark. Secondly the sender has total control over the encryption which was not possible in the traditional encryption methods. And lastly the integrity of the original audio signal stays preserved and does not get out of sync or phase, so if we want to send the audio file to a 3rd party we can do so with the watermarked audio intact.



Contents

- clear memory & command window
- load data
- prepare host
- extract watermark
- show image

clear memory & command window

```
clc
clear
close all
```

load data

```
wm_sz     = 20000;                      % watermark size
px_sz     = wm_sz/8;                    % number of pixels
im_sz     = sqrt(px_sz);                % image size
host_new  = audioread ('host_new.wav');   % new (watermarked) host signal
host_new  = uint8(255*(host_new + 0.5)); % double [-0.5 +0.5] to 'uint8' [0 255]
```

prepare host

```
host_bin  = dec2bin(host_new, 8);        % binary host [n 8]
```

extract watermark

```
wm_bin_str = host_bin(1:wm_sz, 8);
wm_bin    = reshape(wm_bin_str, px_sz , 8);
wm_str    = zeros(px_sz, 1, 'uint8');
for i     = 1:px_sz                       % extract water mark from the first plane of host
wm_str(i, :) = bin2dec(wm_bin(i, :));     % Least Significant Bit (LSB)
end
wm        = reshape(wm_str, im_sz , im_sz);
```

show image

```
imshow(wm)
```

Figure 4.9.7 - HTML output

CHAPTER 5

FINAL RESULT

5.1 Result Discussion

In today's world we know the importance of encryption and privacy and with data being the most prized possession it is more important than ever to protect that data. Therefore by keeping that as our motivation in our minds we started this project. And finally after the completion of this project we have found that using digital watermarking for enhanced signal protection is the most convenient way to target the masses and make audio encryption a better part of everybody's lives. That is why after analyzing 20+ literature reviews 5+ algorithms in the field of audio encryption we came up with the idea of digital watermarking. And we are proud to present the result of our experiment for the same where we have seen that digital watermarking strikes a perfect balance between audio encryption and capital spent/encryption. Therefore to lay bare the final results of our work in a comprehensive form we have summarized the gist of the above work of 40+ pages into the table as the result of our work:

Table 5.1 - Result Comparison

TESTING PARAMETERS	Digital Watermarking Of Audio Signal	Discrete Wave Transform Method	Quantum Discrete Cosine Transform Method	Traditional Overlapping Signal Tone Method
High system specs. required?	NO	YES	YES	NO
Additional hardware required?	NO	YES	NO	NO
Requires integration of AI and ML?	NO	NO	YES	NO
Fluctuating results?	NO	NO	YES	NO
Damage to the original audio?	NO	NO	YES	YES
Receiver's needs integrated?	YES	YES	NO	NO
Dynamic real time encryption?	YES	YES	NO	NO
Can handle complex files?	NO	YES	YES	NO
Additional security can be added?	NO	YES	YES	NO

CHAPTER 6

CONCLUSION

6.1 FUTURE SCOPE

The method " Digital watermarking of Audio Signals" was found more efficient and reliable to encrypt the audio signals as it retained the original audio quality. Furthermore it is convenient to use as it contains simple MATLAB coding. The only criterias being the image is confined to the .png and .jpeg extension and the image size should be greater than the host file i.e the original audio. Not only is this method unique from other proposed methods but also this would pave the way for many researchers to do advanced research in the near future.

REFERENCES

[1] "Change Topic: Pseudorandom Noise (PRN) Expansion" (PDF). GPS.GOV. Retrieved July 13, 2011.

[2] Public Domain This article incorporates public domain material from the General Services Administration document: "Federal Standard 1037C". (in support of MIL-STD-188)

[3] Pi seems a good random number generator – but not always the best, Chad Boutin, Purdue University

[4] Kendall, M.G.; Smith, B. Babington (1938). "Randomness and Random Sampling Numbers". Journal of the Royal Statistical Society. 101 (1): 147–166. doi:10.2307/2980655. JSTOR 2980655.

[5] Yongge Wang. Statistical Testing Techniques For Pseudorandom generation.

[6] Yongge Wang: On the Design of LIL Tests for (Pseudo) Random Generators and Some Experimental Results. PDF

[7] Wang, Yongge; Nicol, Tony (2015). "Statistical Properties of Pseudo Random Sequences and Experiments with PHP and Debian OpenSSL". Computers and Security. 53: 44–64. doi:10.1016/j.cose.2015.05.005.

[8] Knuth, Donald (1998). The Art of Computer Programming Vol. 2 : Seminumerical Algorithms. Addison Wesley. pp. 93–118. ISBN 978-0-201-89684-8.

[9] N. Suryana, Siaw-Lang Wong (2010). “An efficient compact Tchebichef Moment for image compression”: Information Sciences Signal Processing and their Applications (ISSPA), 2010 10th International Conference on

[10] S. E. Tsai and S. M. Yang, An Effective Watermarking Method Based on Energy Averaging in Audio Signals: Hindawi Mathematical Problems in Engineering Volume 2018

[11] Laurence Boney, Ahmed H. Tewk and Khaled N. Hamdy, Digital Watermarks for Audio Signals, Département Signal, Department of Electrical Engineering, University of Minnesota: Minneapolis, MN 55455.

[12] M. Yamini, H.Karmouni, M.Sayyouri, Efficient watermarking algorithm for digital audio/speech signal, Engineering, Systems and Applications Laboratory, National School of Applied Sciences, Sidi Mohamed Ben Abdellah University, BP 72, My Abdallah Avenue Km. 5 Imouzzer Road, Fez, Morocco.

[13] F.Benedetto,G.Guinta, A.Neri, Digital audio watermarking for QoS assessment of MP3 music signals, Dept. of Applied Electronics, University of ROMA TRE, Rome, Italy.

[14] Mohsen Yousefi Nejad, Mohammad Mosleh,A blind quantum audio watermarking based on quantum discrete cosine transform, Department of Computer Engineering, Tabriz Branch, Islamic Azad University, Tabriz, Iran.

[15] Amita Singha, Muhammad Ahsan Ullah, Development of an audio watermarking with decentralization of the

[16] Watermarks, Department of Computer Engineering, Tabriz Branch, Islamic Azad University, Tabriz, Iran

[17] "QoS - End-to-end Quality of Service support over heterogeneous networks". Project website. 2004–2006. Archived from the original on April 30, 2007. Retrieved October 12, 2011.

[18] IPSphere: Enabling Advanced Service Delivery Archived January 13, 2011, at the Wayback Machine

[19] "End-to-end quality of service support over heterogeneous networks". Project description. European Community Research and Development Information Service. Retrieved October 12, 2011.

[20] Torsten Braun; Thomas Staub (2008). End-to-end quality of service over heterogeneous networks. Springer. ISBN 978-3-540-79119-5.

[21] "Multi Service Access Everywhere (MUSE)". Project website. Retrieved October 12, 2011.

[22] "Multi Service Access Everywhere". Project description. European Community Research and Development Information Service. Retrieved October 12, 2011.

[23] Nasir Ahmed; T. Natarajan; Kamisetty Ramamohan Rao (January 1974). "Discrete Cosine Transform" (PDF). IEEE Transactions on Computers. C-23 (1)

[24] J. P. Princen, A. W. Johnson und A. B. Bradley: Subband/transform coding using filter bank designs based on time domain aliasing cancellation, IEEE Proc. Intl. Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2161–2164, 1987.

[25] Schroeder, Manfred R. (2014). "Bell Laboratories". Acoustics, Information, and Communication: Memorial Volume in Honor of Manfred R. Schroeder. Springer. p. 388. ISBN 9783319056609.

[26] Bailey, David H.; Swarztrauber, Paul N. (1994), "A fast method for the numerical evaluation of continuous Fourier and Laplace transforms" (PDF), SIAM Journal on Scientific Computing, 15 (5): 1105–1110, CiteSeerX 10.1.1.127.1534, doi:10.1137/0915067.

[27] Boashash, B., ed. (2003), Time–Frequency Signal Analysis and Processing: A Comprehensive Reference, Oxford: Elsevier Science, ISBN 978-0-08-044335-5.

[28] Bochner, S.; Chandrasekharan, K. (1949), Fourier Transforms, Princeton University Press.

[29] Bracewell, R. N. (2000), The Fourier Transform and Its Applications (3rd ed.), Boston: McGraw-Hill, ISBN 978-0-07-116043-8.

[30] Campbell, George; Foster, Ronald (1948), Fourier Integrals for Practical Applications, New York: D. Van Nostrand Company, Inc..

[31] Champeney, D.C. (1987), A Handbook of Fourier Theorems, Cambridge University Press.

[32] Chatfield, Chris (2004), The Analysis of Time Series: An Introduction, Texts in Statistical Science (6th ed.), London: Chapman & Hall/CRC, ISBN 9780203491683.

[33] Clozel, Laurent Delorme, Patrice (1985), "Sur le théorème de Paley-Wiener invariant pour les groupes de Lie réductifs réels", Comptes Rendus de l'Académie des Sciences, Série I, 300: 331–333.

[34] Condon, E. U. (1937), "Immersion of the Fourier transform in a continuous group of functional transformations", Proc. Natl. Acad. Sci., 23 (3): 158–164, Bibcode:1937PNAS...23..158C, doi:10.1073/pnas.23.3.158, PMC 1076889, PMID 16588141.

[35] Tanguiane (Tangian), Andranick (1993). Artificial Perception and Music Recognition. Lecture Notes in Artificial Intelligence. Vol. 746. Berlin-Heidelberg: Springer. ISBN 978-3-540-57394-4.

[36] Tanguiane (Tanguiane), Andranick (1994). "A principle of correlativity of perception and its application to music recognition". Music Perception. 11 (4): 465–502. doi:10.2307/40285634.

[37] Benjamin Teitelbaum, Stanislav Shalunov (May 3, 2002). "Why Premium IP Service Has Not Deployed (and Probably Never Will)". Draft Informational Document. Internet2 QoS Working Group. Archived from the original on August 30, 2002. Retrieved October 15, 2011.

[38] Andy Oram (June 11, 2002). "A Nice Way to Get Network Quality of Service?". Platform Independent column. O'Reilly. Archived from the original on August 5, 2002. Retrieved October 15, 2011.

[39] Gary Bachula (February 7, 2006). "Testimony of Gary R. Bachula, Vice President, Internet2" (PDF). pp. 2–3. Archived from the original (PDF) on January 7, 2010. Retrieved October 15, 2011.

[40] "X.641: Information technology - Quality of service: framework". ITU-T Recommendation. December 1997.