IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 18, Issue 1, Ver. I (Jan – Feb. 2016), PP 49-56
www.iosrjournals.org

An Enhanced Scheme for Hiding Text in Wave Files

Buthainah F. AL-Dulaimi[1], Nibras Amer Mohammed[1], Hamza A. Al-Sewadi[2]
[1](*Computer Department, College of Education for Women/ University of Baghdad, Iraq)*
[2](*Computer Science Department, King Hussein Faculty of Computing/ Sumaya University for Technology, Amman, Jordan)*

Abstract: *Steganography is a term applied to any number of processes that embed an object into another object in order to deceive any observer or adversary. An embedding algorithm for hiding messages into wave files or audio signals is proposed here. It a spatial domain type that implement least significant bit (LSB) process and represents a jumping algorithm technique.*
Keywords: *Steganography, Information Hiding, LSB, PSNR, Data Security.*

I. Introduction

Two kinds of techniques were used to conceal information nowadays, namely: Encryption and information hiding. The importance of these two protection methods is vastly increased since the discovery of telephone, fax, electronic communication and computer [1]. However, it is still 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 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 [2].

Steganograghy is the practice of hiding private or sensitive information within something that appears to be nothing out of usual. Staganograghy is often confused with cryptology because they are similar in the way that they are both used to protect important information. The main difference between the two is that Steganograghy involves hiding information so it appears as no information is hidden at all. If a person or persons views the carrier object that the information is embedded inside it, he/she will have no idea that there is any hidden information; therefore there will no attempt to decrypt the information [3]. The most common use of Steganograghy is to hide a file inside another file. When information or a file is hidden inside a carrier file, the data is usually encrypted with a key or a password. Steganography literally means "covered writing", which aims to transmit a message through a channel where some other kinds of information are already being transmitted [4]. The general principle of steganography is illustrated in the block diagram of Fig 1 [2] [3].

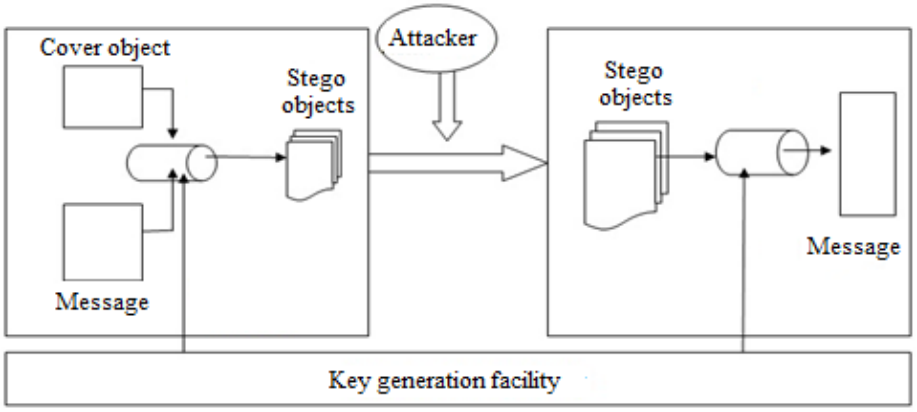

Fig 1. General scheme of steganography

II. Steganography Concepts

Generally, there are numerous methods for hiding information inside another media. Both the information and the media types can be text, Audio or Video files [2]. According to the type of cover media the steganography techniques may be classified as listed below.

Embedding in still Images

Data hiding in still images presents a variety of challenges that arise due to the way the human visual system (HVS) works and the typical modifications that the images undergo [5]. 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 images which are: LSB insertion, Spread spectrum, Texture block, Patchwork, Orthogonal projection coefficients manipulation and other methods like dithering manipulation, perceptual masking, and DCT coefficients manipulation [5].

When hiding information inside images in spatial domain, the LSB (Least Significant bit) method is usually used. The best type of image file to hide information inside of is a 24 bit BMP (Bitmap) image. The reason being is that it is with the highest quality and obviously has largest file size. When an image is of high quality and resolution it is a lot convenient to hide and mask information inside it. Although 24 bit images are best for hiding information inside them due to their size some people may choose to use 8 bit BMP's or possibly another image format such as GIF. The reason for such choice is to avoid suspicion when large images are posted on the internet. Moreover, it is important to remember that if information is embedded inside an image file and that file is converted to another image format, the hidden information is most likely would be lost [5].

2.2 Embedding in Video files

Video files are generally a sequence of images and sounds. The great advantages of videos are the large amount of data that can be hidden inside and the fact that it is a moving stream of images and sounds. Hence, any small but otherwise noticeable distortions might go unobserved by human eyes because of the continuous flow of information [6].

2.3 Embedding in Audio files

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 [3].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 [5].

III. Digital Audio Structure

Digital Audio Files consist of two main parts; header and audio data. The header is used to store information about the file, including the resolution, sampling rate and type of compression. Its structure is based on chunks and sub-chunks. Each Chunk has a type, represented by a number of characters called Chunk ID, comes first in the chunk, followed by the Chunk Size(long integer) which is the size of the rest chunk data (except the size of Chunk ID, and Chunk Size), then comes the content of the chunk [7]. The Resource Interchange File Format (RIFF) provides a way to store all varied data types [8]. The first thing in the contents of the RIFF chunk is the form type which describes the overall type of the files contents such as WAV, AVI, etc [8]. There are two sub-chunks that are required in order to successfully save sampled audio wave formats [7]; (1) the format chunk, specifying the data format and (2) the data chunk, containing the actual sample data.

The main advantage of using this chunk structure is that when parsing a wave file, one doesn't need to interpret every chunk type but he/she could skip over the not needed sub-chunks by reading the Chunk ID and Chunk Size parameters. After identifying the type of the sub-chunk (using Chunk ID) and discovering it is not the wanted one, one can skip this chunk by using the second parameter. The size of this sub-chunk usually equals 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-Chunk ID field consists of four characters (4 bytes) and the field of sub-Chunk size is long integer type (4 bytes) type, hence in case of the existence of the additional two bytes, they should be by-passed [6]

IV. Least Significant Bit Manipulation

LSB is an extremely simple steganographic method that considers embedding in individual pixels of the cover image. Each of these pixels in cover is made up of a string of bits. One may commandeer the use of the 2-least significant bits of 8-bit true color image to hold 2-bits of the message to be embedded by simply overwriting the data that was already there. Empirically, it is noted that the impact of changing the 2-least significant bits is almost always entirely imperceptible [9].

4.1 Embedding - Altering/Replacing the LSB

When files are created there are usually some bytes in the file that aren't really needed, or at least aren't that important. These areas of the file can be replaced with the information that is to be hidden without significant altering or damaging to the file. This allows a person to hide information in the file and make sure that no human could detect the change in the file. The LSB method works best for high resolution and multicolor image files. And it is also suitable for audio files that have many different sounds and of a high bit rate [6]. The LSB method usually does not increase the file size, but depending on the size of the embedded information, the file may become noticeably distorted.

Figure 2 illustrates the general description of replacing 2 least significant bits. The message may be a few thousand bits (often at 7 or 8 bits per text character) embedded in millions of other bits. Probably the most typical use is Audio files as carriers. Audio information is commonly stored in either 24-bit or 8-bit files. If an 8-bit Audio is viewed as a grid and the grid is made up of cells, these cells are called pixels. Each pixel consists of an 8-bit binary number (or a single byte), and each 8-bit binary number refers to the color pixel.

4.2 Extraction - Reconstruction of the Secret message

At the receiving side of the stego-audio, the secret message is reconstructed by reversing the embedding process, i.e. extracting the least significant bits from the pixels in exactly the same way as they were embedded. The secret message will be recovered; however the audio file cover will not be restored. Fig 2 illustrate the extraction principle [8].

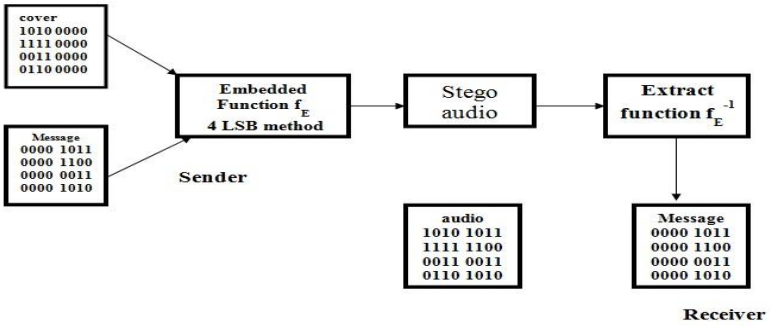

Fig 2.Shows the general description of 4 least significant bits

V. The Proposed Steganographic Scheme

The proposed system consists of two phases. The first phase is to hide the binary text into the wave file and the second phase is to extract the hidden text from the wave file. The general structure of the proposed system is illustrated in Fig 3. It consists of two basic modules: Embedding and extraction modules. The inputs 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 enters to extraction stage is processed through a set of operations to retrieve the secret data. Each module (i.e. embedding and extraction) consists of several steps that are executed systematically in order to produce the final result, and will be described as below.

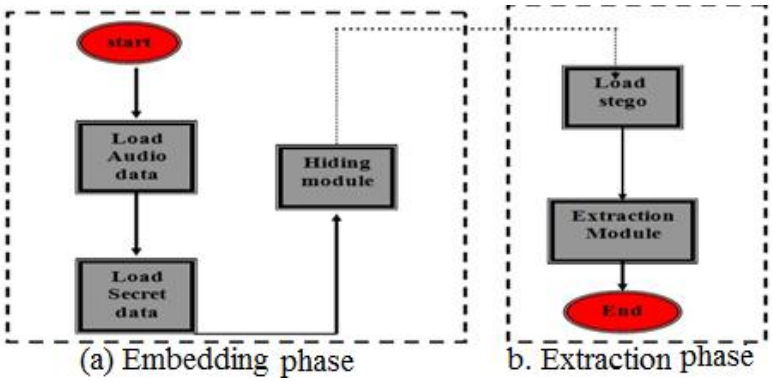

Fig 3. The general steganography model

5.1 Load Audio Data

The audio file that will be used as a carrier and the file to be embedded (secret message) are loaded first. Then they are converted to binary representation to be used for the hiding process. Each character of the message is of 8-bits length. The conversion process is carried out using algorithm (1) shown in Fig 4.

Algorithm (1): 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^A j)
Set K «—K + 8 End loop
Set ScrtLen «— OrgScrtLen x 8
End.

Fig 4. Algorithm (1): Binary conversion.

5.2 Load the Secret Data

After converting the secret message to binary, the binary text is prepare to be embedding into the carrier which is a wave file. This operation of secret file loading is depicted by algorithm (2) shown in Fig 5.

Algorithm (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.

Fig 5. Algorithm (2): Loading the secret message.

5.3 The Hiding Phase

In this phase, the embedding of secret block is done on the wave file (voiced blocks), exclusively. So, the implementation of process (to prepare the slack space) is executed on the voiced blocks. The embedding process is done by adding or subtracting a value, (A) 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 (A) should be less than half the value of quantization step, in order to avoid the occurrence of a jump from current quantization bin to the one of the adjacent (pervious or next) quantization bins. To make the process of determining the suitable values of (A) more easy and consistent, a parameter called slack step ratio (R) is proposed, it is a ratio parameter define by equation 1.

$$R=\Delta/Q \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad (1)$$

The value of R should be chosen in the range ($0 < R < 0.5$) for the reason stated previously. In this research work, the value of R was chosen to be either (1/3) or (1/4). Both the values of (Q) and (R) are predefined by the user.

The quantity (A) is added to phase (pht) if the secret data value (Scrt) is 1, or it is subtracted if (Scrt) is 0, as follows:

$$PH_H(u) = phs_q(u) - \Delta \ , \ ifScrt(i) = 0$$
$$= phs_q(u) + \Delta \ , \ ifScrt\{i) = l \quad . \quad . \quad . \quad . \quad .(2)$$

$Where \ phs_q(u)$ is the j^{th} 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 =l,2,3, . . . , (N/2)-l.

The hiding phase algorithm is illustrated in Fig 6.

Algorithm (3): 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, 21, 32
Str: Start block position of Overlnfo bits vector inCov. Ed: Ending block position of Overlnfo bits vector in Cov.
Output:
Steg: Array of stego wave data of length DataSize.
Step 1: Divide Covinto blocks of size N samples.
step 2:check if (block position<str) or (block position>Ed) then go to Step 9.
step 3: check each block is it unvoiced block, if it is unvoiced go to step 8.
Step 4: Apply dynamic located jump on this unvoiced block to produce B, and BB, Coefficients then construct Phs and Mag .
Step 5: Insert secret bits Scrt in the quantized phase coefficients.
step 6: Reconstruct F ,and Fi coefficients and pass the result through IDFT to construct the stego block, then round the values to the range [0 … 255].
step 7: put the produced block in Steg array, which represents the stego data.
Step 8: If there are (secret bits of Scrt And cover block) go to Step 2.
Step 9: End.

Fig 6. Algorithm (3): The hiding stage.

5.4 The Extraction Stage

The extraction stage is similar to that mentioned 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 equation 3.

$$Rscrt(i) = 0, if\ phs_q(u) > phs(u)$$
$$= 1\ otherwise \quad \ldots\ldots (3)$$

$Where\ phs(u)$ 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*retrieved secret data, and u=l,2, ... ,N\2.The main steps of the extraction stage are illustrated in Fig 7.

Algorithm (4): Extraction Stage
Input:
Steg: Array of stego wave data of length DataSize. Str: Start block position of Overlnfo bits vector in steg. Ed: Ending block position of Overlnfo bits vector in steg
Output:
Scrt: Secret bits of length ScrtSize.
Stepl: Divide Steg into block of size N samples.
Step2 : Check if (block position<Str) or (block position>Ed) then go to step7.
Step 3:Check each block whether it is voiced or unvoiced block, if it is unvoiced jump to step6.
Step 4: Apply in LSB dynamic located jump on the(voiced block to produce sss and bb, coefficients then Construct phs and Mag.
Step 5: Retrieve secret bits Scrt from phs_q .
Step 6: If(retrieve secret bits<ScrtSize)
If block of Steg not finished then jump to step 2.
Else stop with error message indicating that stego file is corrupted.
Step 7: End.

Fig 7. Algorithm (4): the extraction stage.

VI. Implementation and Results

The algorithms of the proposed scheme prototype are coded and run on a Pentium PC. In the following, a summary of an implementation example is first listed together with some samples screens showing the process of both hiding text message into wave files, then extracting the message. Secondly, measurements of the quality of hiding and extraction process are included by calculating the root mean square error (RME) and the Peak signal to noise ratio (PSNR).

6.1 System implementation:

After entering the name and password of the user at the start of running the system, the menu screen of Fig 8 shows the operation action parameters, which consists of the following commands.

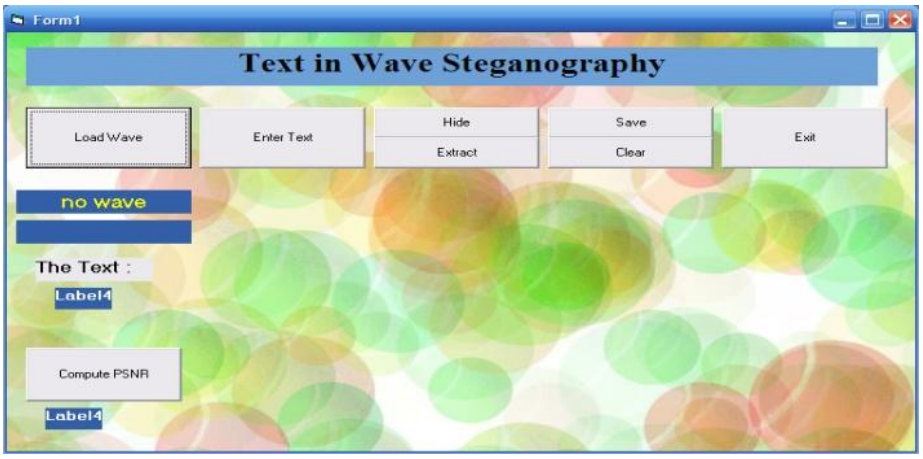

Fig 8. The operation action interface window.

- **Load Wave File**:This command is used to load the cover file which it is an audio file in a wave form type as shown in Fig 9.

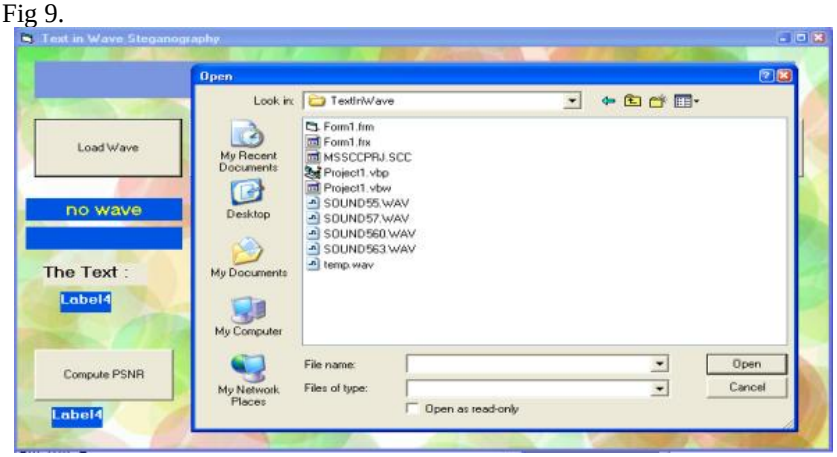

Fig 9. File loading window.

- **Enter Text To Be Hidden**: This command is used to enter the secret messages which are to be hidden as shown in Fig 10. In this system, the secret message is treated as a binary file, and will appear in the text box (e.g. "computer" in this example).

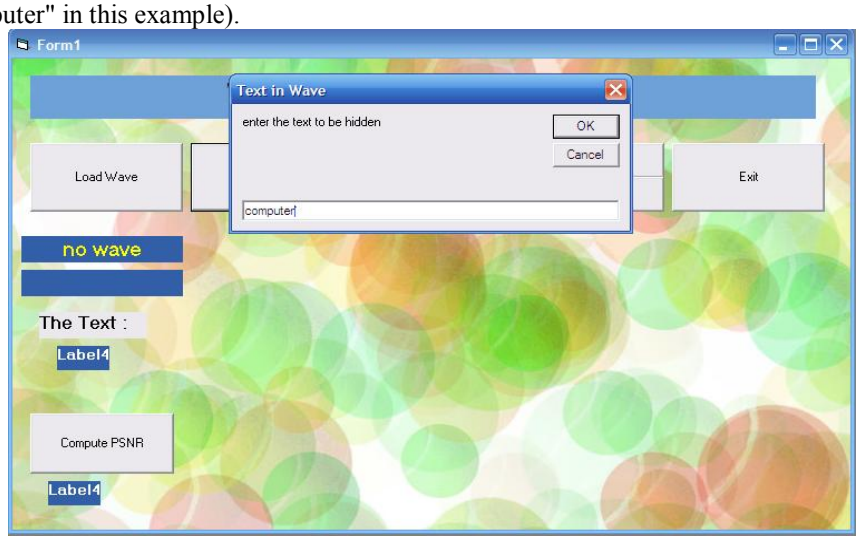

Fig 10. Entering the secret message.

- **Hide Text In Wave File:** This command is the last command that is executed to hide a message which is performing the proposed algorithms in order to hide the secret message into audio file and at the end of this process a confirming comment will appear on the screen informing of the completion of the process.

- **Save Hiding File:** Finally after all previous commands were executed; the resulting file can be saved by this command, specifying its name and destination.
- **Extract The Secret Message:** This command is the inverse of hiding command. The secret message is extracted from the stego-file. It is restored to its original form by implementing the extraction algorithm of Fig 7.
- **Clear Screen:** This command is used to clear the program and the screen from any message. Then it will be ready for a new hiding operation.

6.2 Stabile System Performance.

To talk about the stability of the system performance, it is required to define the peak signal to noise ration first (PSNR).

PSNR: The phrase peak Signal-to-Noise Ratio, often abbreviated **PSNR**, is an engineering term for the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. Because many signals have a very wide dynamic range, PSNR is usually expressed in terms of the logarithmic decibel scale. The PSNR is most commonly used as a measure of quality of reconstruction of loss compression codecs (e.g., for image compression). The signal in this case is the original data, and the noise is the error introduced by compression. When comparing compression codecs it is used as an *approximation* to human perception of reconstruction quality, therefore in some cases one reconstruction may appear to be closer to the original than another, even though it has a lower PSNR (a higher PSNR would normally indicate that the reconstruction is of higher quality). One has to be extremely careful with the range of validity of this metric; it is only conclusively valid when it is used to compare results from the same codec (or codec type) and same content [10]. It is most easily defined via the mean squared error (*MSE*). Given a noise-free *m*×*n* monochrome image *I* and its noisy approximation *K*, *MSE* is defined by equation 4.

$$MSE = \frac{1}{m\,n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2 \quad \ldots\ldots (4)$$

The PSNR is defined by equation 5.

$$\begin{aligned} PSNR &= 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) \\ &= 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \\ &= 20 \cdot \log_{10}(MAX_I) - 10 \cdot \log_{10}(MSE) \quad \ldots (5) \end{aligned}$$

Here, MAX_I is the maximum possible pixel value of the image. When the pixels are represented using 8 bits per sample, this is 255. More generally, when samples are represented using linear PCM with *B* bits per sample, MAX_I is 2^B-1. For color images with three RGB values per pixel, the definition of PSNR is the same except the MSE is the sum over all squared value differences divided by image size and by three. Alternately, for color images the image is converted to a different color space and PSNR is reported against each channel of that color space, e.g., YCbCr or HSL [9].Typical values for the PSNR in lossy image and video compression are between 30 and 50 dB, where higher is better. Acceptable values for wireless transmission quality loss are considered to be about 20 dB to 25 dB [10].

Calculations of the PSNR for listed in table I. It shows the calculation results for more than one case (e.g. for cover audio files of different sizes, namely; 22 KB, 30 KB, and 956 KB). It is notices that PSNR values are reduced as the size of the embedded message is increased, i.e. the results are generalized for all cases. However, the proposed method has experienced a good stability in performance for different cases studied.

Table 1. The Psnr Measurement For The Proposed Scheme.

Case Stabile	Cover Audio File Size	Secret Message Size	PSNR
1	**22 KB**	8 Byte	13.5023767966667
		34 Byte	10.4597538329787
		47Byte	9.7054155119719
2	**30 KB**	8 Byte	13.0846465394163
		34 Byte	10.2454903333924
		47Byte	9.53434500931639
3	**958 KB**	8 Byte	44.4773998751594
		34 Byte	40.322785503722
		47Byte	34.81240623547805

VII. Conclusion

The implementation and measurement of the given algorithm have manifested some satisfactory results from which one can conclude the following.

1. Data or text 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.
2. Large threshold value provide more power in cover audio signal by avoiding unvoiced blocks and increased correct retrieved bits, but also decrease in hiding rate.
3. Whenever the size of the secret message increases, the value of PSNR is decreases. This characteristic is notices for the various cover audio files that were included in the test.

REFERENCES

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

[2]. N. F. Johnson, and Jajodia, S., Stganalysis of Image Created Using Current Steganography Software, Workshop on Information hiding proceeding, center for secure information System, lecture notes in computer science, 1525, George Mason University, USA, 1998. http://isse.gmu.edu/~csis

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

[4]. R. Petrovic, Winograd, J. M., Jemili J., and Metois, E., Data Hiding Within Audio Signal, Journal of Electronics and Energetics,12(2), 1999, 103-122.

[5]. W. Bender, Techniques for Data Hiding, IBM System Journal; 35(3-4), 1996, 1-10. http://www.research.ibm.com/journal/sj/353/section/bendeaut.htm1#bender

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

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

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

[9]. D. Salomon, 2007, Data Compression: The Complete Reference http://books.google.com/books?id=ujnQogzx_2EC&lpg=PA281&ots=FolwqB8qsN&dq=PSNR%20infinite&pg=PA281#v=onepage&q=PSNR%20infinite&f=false

[10]. Q. Huynh-Thu, Ghanbari M., Scope of Validity of PSNR in Image/Video Quality Assessment, Electronics Letters, 2008.