ISSN No. 0976-5697

Volume 2, No. 3, May-June 2011

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Information Hiding in Higher LSB Layer in an Audio Image

Prof. Samir Kumar Bandyopadhyay*
Dept. of Computer Sc. & Engineering,
University of Calcutta 92 A.P.C. Road,
Kolkata , India
skb1@vsnl.com

Biswajita Datta
Lecturer,
Department of Computer Science & Engineering.
St. Thomas College of Engineering & Technology
Kolkata , India
biswa.jita@gmail.com

Koushik Dutta
Student,
Department of Information Technology.
Murshidabad College of Engineering & Technology
Kolkata , India
koushik.it.22@gmail.com

Abstract: With the rapid advancement of Communication Technology information transformation through internet become very much popular as well as easy and less time consuming. But this type advancement in the field of data communication in other sense has hiked the fear of getting the data snooped at the time of sending it from the sender to the receiver. Information protection is an issue in rapidly evolving contemporary information technologies. Steganography plays an important role in the field of information security by hiding messages in such a way that no one apart from the sender and intended recipient realizes the existence. Here in this paper we propose a novel audio steganographic method for embedding information. For embedding here we use higher LSB layers for increasing robustness. The selected amplitudes are modified in such a way that they can contain two bits without affecting the perceptual quality. In this paper we propose a new compression technique for sending compressed information.

Keywords: Steganography, Human Auditory System (HAS), ASCII, Cover audio, Target data, Stego - audio, Encoding, Decoding

I. INTRODUCTION

Information protection is a topical issue in rapidly evolving contemporary information technologies. The need to ensure that only the right people have authorization to high-security accesses, has led to the development of information hiding. So it is desired that the communication to be done in secrete. Such secrete communication ranges from the obvious cases of bank transfers, corporate communications and credit card purchases, on down to a large percentage of everyday email. Steganography can be an effective means that enables concealed data to be transferred inside of seemingly innocuous carrier files. Generally speaking, steganography brings science to the art of hiding information. The purpose of steganography is to convey a message inside of a conduit of misrepresentation such that the existence of the message is both hidden and difficult to recover when discovered.

Steganography is the art and science of writing hidden messages in such a way that no one, apart from the sender and intended recipient, suspects the existence of the message, a form of security through obscurity. The word steganography is derived from the Greek words "*stegos*" meaning "cover" and "*grafia*" meaning "writing" defining it as "covered writing". The term "steganography" came into use in 1500's after the appearance of Trithemius' book on the subject "Steganographia". Its ancient origins can be traced back to 440 BC. Although the term steganography was only coined at the end of the 15th century, the use of steganography dates back several millennia. In ancient times, messages were hidden on the back of wax writing tables, written on the stomachs of rabbits, or tattooed on the scalp of slaves - all these incidents were the earliest known examples of what we now a days known as Steganography.

Modern steganography is generally understood to deal with electronic media rather than physical objects. What Steganography essentially does is exploit human perception; human senses are not trained to look for files that have information hidden inside of them. Generally, in steganography, the actual information is not maintained in its original format. Information can be hidden inside a multimedia object using many suitable techniques. As a cover object, we can select image, audio or video file even typeset text. Depending on the type of the cover object, definite and appropriate technique is followed in order to obtain security. The most popular data formats are .bmp, .doc, .jpeg, .mp3, .txt and .wav.

In a pure steganography framework, the technique for embedding the message is unknown to anyone other than the sender and the receiver. An effective steganographic scheme should posses the following desired characteristics [1-2].

Secrecy: a person should not be able to extract the covert data from the host medium without the knowledge of the proper secret key used in the extracting procedure.

Imperceptibility: the medium after being embedded with the covert data should be indiscernible from the original medium. One should not become suspicious of the existence of the covert data within the medium.

High capacity: the maximum length of the covert message that can be embedded should be as long as possible.
Resistance: the covert data should be able to survive when the host medium has been manipulated, for example by some lossy compression scheme [3].
Accurate extraction: the extraction of the covert data from the medium should be accurate and reliable.

Audio Steganography is basically a process of hiding a data, image or an audio clip within an audio file in such a way so that the audio will sound unchanged to Human Auditory System(HAS) and no one, apart from the sender and intended recipient can make out the existence of data, image or audio within the audio file. The human auditory system (HAS) perceives sounds over a range of power greater than 109:1 and a range of frequencies greater than 103:1. The sensitivity of the HAS to the Additive White Gaussian Noise (AWGN) is high as well; this noise in a sound file can be detected as low as 70 dB below ambient level. On the other hand, opposite to its large dynamic range, HAS contains a fairly small differential range, i.e. loud sounds generally tend to mask out weaker sounds [4]. Additionally, HAS is insensitive to a constant relative phase shift in a stationary audio signal and some spectral distortions interprets as natural, perceptually non-annoying ones.

II. RELATED STUDY

One of the more common approaches to substitution is to replace the least significant bits (LSBs) in the cover file (Katzenbeisser, 2000). This method is probably the easiest way of hiding information in an image and yet it is surprisingly effective and it also good for audio and video steganography. It works by replacing the least significant bits of amplitude of cover audio with the bits of target data. On average, LSB requires that only half the bits in an audio be changed. We can hide data in the least and third least significant bits which makes the process perceptually transparent.

The main advantage of the LSB coding method is a very high watermark channel bit rate and a low computational complexity of the algorithm, while the main disadvantage is considerably low robustness against signal processing modifications. The perceptual quality of watermarked audio is higher in the case of the proposed method than in the standard LSB method is implementing in the Electronic Copyright Management System.

As the number of used LSBs during LSB coding increases or, equivalently, depth of the modified LSB layer becomes larger, probability of making the embedded message statistically detectable increases and perceptual transparency of stego objects is decreased. Therefore, there is a limit for the depth of the used LSB layer in each sample of host audio that can be used for data hiding. Robustness of the steganography using the LSB coding method increases with increase of the LSB depth used for data hiding.[8] Therefore, improvement of steganographic robustness obtained by increase of depth of the used LSB layer is limited by perceptual transparency bound, which is the fourth LSB layer for the standard LSB coding algorithm.[9]

We developed a novel method that is able to shift the limit for transparent data hiding in audio using a three step approach. In the first step, a hidden bit is embedded into the i^{th} LSB layer of the host audio using a novel LSB coding method. In the second step, the impulse noise caused by watermark embedding is shaped in order to change its white noise properties. Then in the third step the 1st LSB layer is replaced by next hidden bit.

The standard LSB coding method simply replaces the original host audio bit in the i^{th} layer (here i is 3) with the bit from the hidden bit stream. In the case when the original and hidden bit are different and i^{th} LSB layer is used for embedding the error caused by embedding is 2^i-1 quantization steps (QS) (amplitude range is [-256 to 255]). The embedding error is positive if the original bit was 0 and watermark bit is 1 and vice versa. Our proposed LSB algorithm causes minimal embedding distortion of the host audio. For example, if the original sample value was $00000100_2=4_{10}$, and the watermark bit is zero is to be embedded into 4th LSB layer, instead of value $00000000_2=0_{10}$, that would the standard algorithm produce, the proposed algorithm produces sample that has value $00000011_2=3_{10}$, which is far more closer to the original one. After that if we place 0 at 1st LSB layer the value become $00000010_2=2_{10}$ or if we place 1 at 1st LSB layer then there is no change (i.e. $00000011_2=3_{10}$), which is also closer to the original one. So our proposed technique has a high steganographic capacity (i.e. how many bits can we hide in an image using LSB techniques without causing statistically significant modifications). However, the extraction algorithm remains the same; it simply retrieves the watermark bit by reading the bit value from the predefined LSB layer in the watermarked audio sample.

Some of the most used audio steganographic techniques are Lossless Adaptive Digital Audio Steganography [5], LSB based Audio Steganography [6], Audio Steganography using bit modification [7] etc.

III. PROPOSED METHOD

As the number of used LSBs during LSB coding increases or, equivalently, depth of the modified LSB layer becomes larger, probability of making the embedded message statistically detectable increases and perceptual transparency of stego objects is decreased. Therefore, there is a limit for the depth of the used LSB layer in each sample of host audio that can be used for data hiding. But robustness of the embedding using the LSB coding method increases with increase of the LSB depth used for data hiding. Therefore, improvement of watermark robustness obtained by increase of depth of the used LSB layer is limited by perceptual transparency bound, which is the third LSB layer for the standard LSB coding algorithm.

Our objective here is to hide some information (text) within an audio clip. We call the text to be hidden as target text and the audio under which they are to be hidden as cover audio. We have to hide information using a new proposed method in such a manner so that the audio will sound unchanged to HAS. The entire method can be sub-divided into two individual methods- first, hiding the data into the audio signal (Encoding) and the sago audio is send to the receiver end. Second receive the stego audio and extracting the hidden data out of it (Decoding). Now we are going to explain them in parts.

IV. ENCODING TECHNIQUE

An audio clip consists of a number of amplitude values in the range -1 to +1 such that the values are like 0.8134,0.0313,-0.0078,-0.0002,0 etc. It is tough to work with this value. So to normalize these values we multiply the amplitude values of the audio file with multiple of 10 so that the amplitude values are lies between -255 to 254 and we can represent them in 8 bits.

Hiding String Length

We then find the length of the target string (store it in the variable 'strlen') that we are going to hide. Then convert it into its 8 bit equivalent. We store this length within the first 8 amplitudes by modifying only the LSB. We can represent this whole process with the help of a simple diagram as shown below.

Suppose the target string is "this is a secret code". Length of the string (strlen) is 21. We represent 21 in 8 bits as follows-

Strlen in 8 bits =

$\boxed{0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1}$ 21

Now the 8 bits are placed according to the following Table I.

Table I

Amplitude Position	Amp. Value in Decimal	Amp. Value in Binary	Modified Amp. Value in Decimal	Modified Amp. Value in Binary
1	10	0 0 0 0 1 0 1 **0**	11	0 0 0 0 1 0 1 **1**
2	99	0 1 1 0 0 0 1 **1**	98	0 1 1 0 0 0 1 **0**
3	21	0 0 0 1 0 1 0 **1**	20	0 0 0 1 0 1 0 **1**
4	52	0 0 1 1 0 1 0 **0**	52	0 0 1 1 0 1 0 **0**
5	20	0 0 0 1 0 1 0 **0**	21	0 0 0 1 0 1 0 **1**
6	22	0 0 0 1 0 1 1 **0**	22	0 0 0 1 0 1 1 **0**
7	20	0 0 0 1 0 1 0 **0**	20	0 0 0 1 0 1 0 **0**
8	29	0 0 0 1 1 1 0 **1**	28	0 0 0 1 1 1 0 **0**

Amplitude Selection

As we can see the change thus obtained in the amplitude value is negligible. But if all the adjacent amplitudes are changed at the same time it may bring about a large change. For this reason instead of changing all the adjacent amplitudes, we change a certain number of selected amplitude. The amplitudes are selected according to the following order. Suppose the starting point is 1, then after the 1st amplitude, the next amplitudes changed will be the following order: -

1	6	15	28	45
[(1+0)*1]	[(2+1)*2]	[(3+2)*3]	[(4+3)*4]	[(5+4)*5]
(13th)	**(19th)**	**(28th)**	**(41st)**	**(58th)**

But if the length of the string be 21, we start it from 21. Then the affected amplitudes are selected according to following order:

21		861
[(21+20)*21]		[(22+21)*22]
946	. . .	3321
[(23+22)*23]	. . .	[(41+40)*41]

Data Hiding Mechanism

In case of data hiding we replace two LSBs – 3rd and 1st of the amplitude values of audio signal with the binary value of ASCII of the character to be hidden. Now we demonstrate our proposed method.

For hiding individual character first we convert them in 7 bit ASCII and then cut the MSB to get 6bits of them. Suppose we want to hide 'A'. The ASCII value of 'A' is 65, in binary which is represented in 7 bit as '1000001'. Before hiding them we discard the MSB '1' of '1000001'and get 6 bit as '000001'. Then cut these bits in 2- 2- 2 order as follows:

$$\underbrace{00} \qquad \underbrace{00} \qquad \underbrace{01}$$

Then each of these two bits is placed in 3rd and 1st LSB. So for hiding a character we require 3 amplitude values. First bit of two bits is hidden in the 3rd LSB of amplitude represented in 8 bits by some modification of the original amplitudes. Then 2nd bit is replaced at 1st LSB position.

Now we discuss how the modification of the original amplitude value is done after the replacing of 3rd LSB of the amplitude of audio signal with the bit of the character.

The adjustment algorithm has three cases-

I. The 3rd bit (a_i) of the sample amplitude is modified from 0 to 1

II. The 3rd bit (a_i) of the sample amplitude is modified from 1 to 0.

III. The 3rd remains the same as the original i.e. we place 1 in place of 1or 0 in place of 0.

Case I:

First we replace 3rd LSB (a_i) of original amplitude which is originally 0 with 1 and then adjust the original amplitude value with any one of the following 4 cases. For Case I there are another 4 cases-

Case A:

a_{i-1}(2nd LSB) = 1 and a_{i+1}(4th LSB) = 1.

Then we put 0 to $a_{i-1}, a_{i-2}, \ldots\ldots, a_0$.

We can explain this with the help of an example. Suppose the original 8 bit amplitude is-

1	0	0	1	1	**0**	1	0
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

154

After changing 3rd LSB from 0 to 1.

				↓		↓	
1	0	0	1	1	**1**	1	0
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

After adjustment the changed amplitude becomes

1	0	0	1	1	**1**	0	0
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

156

Case B:

a_{i-1}(2nd SB) = 1 and a_{i+1}(4th LSB) = 0.

Then we put 0 to $a_{i-1},a_{i-2},\ldots\ldots,a_0$.

We can explain this with the help of an example. Suppose the original 8 bit amplitude is-

1	0	0	0	0	**0**	1	0
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

130

After changing 3rd LSB from 0 to 1

				↓		↓	
1	0	0	0	0	**1**	1	0
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

After adjustment the changed amplitude becomes

1	0	0	0	0	**1**	0	0
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

132

Case C:

a_{i-1}(2nd LSB) = 0 and a_{i+1}(4th LSB) = 1.

Then we put 1 to $a_{i-1},a_{i-2},\ldots\ldots,a_0$ and $a_{i+1} = 0$

We can explain this with the help of an example. Suppose the original 8 bit amplitude is-

1	0	1	0	1	**0**	0	1
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

169

After changing 3rd LSB from 0 to 1

				↓		↓	
1	0	1	0	1	**1**	0	1
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

After adjustment the changed amplitude becomes

1	0	1	0	0	**1**	1	1
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

167

Case D:

a_{i-1}(2nd LSB) = 0 and a_{i+1}(4th LSB) = 0.

Then we put 1 to $a_{i-1},a_{i-2},\ldots\ldots,a_0$ and then we replace all the bits with value 0 towards the MSB by 1 until we encounter 1, when we get a bit towards the MSB with value 1, we simply replace that bit with 0 and stop the process.

We can explain this with the help of an example. Suppose the original 8 bit amplitude is-

1	0	1	0	0	**0**	0	1
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

161

After changing 3rd LSB from 0 to 1

				↓		↓	
1	0	1	0	0	**1**	0	1
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

After adjustment the changed amplitude becomes

1	0	0	1	1	**1**	1	1
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

159

Case II:

First we replace 3rd LSB (a_i) of original amplitude which is originally 1 with 0 and then adjust the original amplitude value with any one of the following four cases. For Case II there are another four cases-

Case A:

a_{i-1}(2nd LSB) = 0 and a_{i+1}(4th LSB) = 0.

Then we put 1 to $a_{i-1},a_{i-2},\ldots\ldots,a_0$.

We can explain this with the help of an example. Suppose the original 8 bit amplitude is-

1	0	0	0	0	**1**	0	0
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

132

After changing 3rd LSB from 1 to 0

				↓		↓	
1	0	0	0	0	**0**	0	0
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

After adjustment the changed amplitude becomes

1	0	0	0	0	**0**	1	1
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

131

Case B:

a_{i-1}(2nd LSB) = 0 and a_{i+1}(4th LSB) = 1.

Then we put 1 to $a_{i-1},a_{i-2},\ldots\ldots,a_0$.

We can explain this with the help of an example. Suppose the original 8 bit amplitude is-

1	0	0	1	1	**1**	0	0
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

156

After changing 3rd LSB from 1 to 0

				↓		↓	

1	0	0	1	1	**0**	0	0	
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	

After adjustment the changed amplitude becomes

1	0	0	1	1	**0**	1	1	155
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	

Case C:

a_{i-1}(2nd LSB) = 1 and a_{i+1}(4th LSB) = 0.

Then we put 0 to $a_{i-1}, a_{i-2}, \ldots\ldots, a_0$ and $a_{i+1} = 1$

We can explain this with the help of an example. Suppose the original 8 bit amplitude is-

1	0	0	0	0	**1**	1	0	134
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	

After changing 3rd LSB from 1 to 0

				↓		↓		
1	0	0	0	0	**0**	1	0	
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	

After adjustment the changed amplitude becomes

1	0	0	0	1	**0**	0	0	136
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	

Case D:

a_{i-1}(2nd LSB) = 1 and a_{i+1}(4th LSB) = 1.

Then we put 0 to $a_{i-1}, a_{i-2}, \ldots\ldots, a_0$ and then we replace all the bits with value 1 towards the MSB by 0 until we encounter 0, when we get a bit towards the MSB with value 0, we simply replace that bit with 1 and stop the process.

We can explain this with the help of an example. Suppose the original 8 bit amplitude is-

1	0	1	1	1	**1**	1	1	191
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	

After changing 3rd LSB from 1 to 0

				↓		↓		
1	0	1	1	1	**0**	1	1	
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	

After adjustment the changed amplitude becomes

1	1	0	0	0	**0**	0	0	192
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	

Case III:

If the bit which we want to place at the 3rd LSB position of the amplitude is same with the 3rd LSB of the original amplitude then the original amplitude value become unchanged. Here we also may think of two cases:

Case A:

a_i(3rd LSB) = 0 and replace with 0.

1	0	0	0	0	**0**	0	0	128
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	

After changing 3rd LSB from 0 to 0

1	0	0	0	0	**0**	0	0	128
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	

Case B:

a_i(3rd LSB) = 1 and replace with 1.

1	0	0	0	0	**1**	0	0	132
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	

After changing 3rd LSB from 1 to 1

1	0	0	0	0	**1**	0	0	132
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	

At first we are mainly concerned about the 3rd LSB. After all the modification of the amplitude we insert the 2nd bit of two bit pair at the 1st LSB position. For Example, if the original value of the amplitude be 132 after LSB replacement with 1 it becomes 133 or with 0 it remain same with 132.

1	0	0	0	0	1	0	**0**	132
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	

1st LSB changes with 1.

1	0	0	0	0	1	0	**1**	133
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	

1st LSB changes with 0.

1	0	0	0	0	1	0	**0**	132
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	

1	0	0	0	0	1	0	**0**	132
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	

V. DECODING TECHNIQUE

In this case the decoding method is very simple. The receiver when receives the stego-audio file, his target is to extract the data hidden in it. First the receiver extracts length of the targeted string by concatenating the LSB bit of first 8 amplitudes. Now according to the above discussed series the amplitudes of the audio signal is selected. Then from each of the selected amplitude value we pick the 3rd LSB and the 1st LSB. To get each character of the targeted string 6 bits(3rd LSB and 1st LSB) from every 3 consecutive amplitudes (2 from each amplitude) that fall in the sequence are picked up. By concatenating them we get 6 bit equivalent of a character of the string. If two MSB of 6bit string be 0,0 or 0,1, then concatenate 1 else if 1,0 or 1,1 then concatenate 0 as MSB with the 6 bit string to make it as 7 bit. Then covert this 7 bit to its decimal equivalent which is the ASCII value of the character.

We can explain this with the help of a diagram-

Suppose the values of the 13th, 19th and 28th amplitudes in 8 bit representation are-

13th amplitude	1	0	0	1	1	**1**	1	**1**
19th amplitude	1	1	0	0	0	**0**	0	**0**
28th amplitude	1	0	0	0	0	**0**	1	**1**

From these 3 consecutive amplitudes that fall in the sequence, we can obtain the 6 bit representation of the character hidden in the 3rd LSB and the 1st LSB of three consecutive amplitudes according to the sequence.
The 6 bit string comes to be-

1	1	0	0	0	1
c_5	c_4	c_3	c_2	c_1	c_0

here ince the two MSB (c_5) and (c_4) are (1,1). So, we concatenate 0 as MSB with the 6 bit representation to obtain the 7 bit equivalent of the character.
The 7 bit string comes to be-

The ASCII value of the character is 49.

0	1	1	0	0	0	1
c_6	c_5	c_4	c_3	c_2	c_1	c_0

Therefore, the decoder obtains 1st character as **1**.
If we follow the above steps for the number of amplitude values that falls in the sequence equal to thrice the length of the string we get the target string at the receiver site from the stego audio.

ENCODING ALGORITHM OF HIDING DATA IN AN AUDIO FILE

```
Step 1: Start
Step 2: Read the Cover Audio file and the target Text
        massage.
Step 3: Normalize the cover audio so that we can represent
        the amplitudes in 8 bit Binary.
Step 4: Convert the target massage into upper Case.
Step 5: Calculate the length of the target String and store it
        in a variable – STRLEN
Step 6: if STRLEN>8 then
            Set Len = STRLEN
        else
            Set Len = 8+STRLEN
        L=Len
Step 7: Store the value of STRLEN in the first eight
        amplitudes of the Cover Audio using the function
        STOR_LEN (STRLEN, Cover Audio).
Step 8: For 1 to STRLEN repeat the steps 9 to 11.
Step 9: Convert the ASCII value of the individual
        character of the target String into its 7 bit binary
        equivalent.
Step 10: Cut the MSB to convert the 7 bit binary into 6 bit
         binary.
Step 11: To store this 6 bit we use three consecutive
         amplitudes according to the series we consider
         here. To do this repeats the step 12 to 16 for 1 to
         6 incremented by 2 for each character. Here each
         time we consider pair of two consecutive bits.
Step 12: Use the function M (L) to find out the amplitude
         location of the cover audio for embedding the
         target message.
Step 13: Replace 3rd LSB ($a_i$) of target amplitude of the
         cover audio by a bit of 6 bit binary equivalent of
         a character and then adjust the amplitude value of
         the targeted cover audio using the function
         ADJUST(modified cover amplitude)
Step 14: Then replace the 1st LSB of the adjusted
         amplitude.
Step 15: Restore the bits of the target amplitude with
         modified values in the cover Audio.
Step 16: L=L+1.
Step 17: Send the stego- audio to the receiver.
Step 18: End
```

Algorithm for Function ADJUST (a)

/* Adjust the value of amplitude ‘a’ after replacing 3rd LSB. */

```
Step 1: If the 3rd bit of the amplitude value ($a_i$) is modified
        from 0 to 1 then
    Step 1.1: If $a_{i-1}$ =1 & $a_{i+1}$=1 then
        Step 1.1.1: Assign $a_{i-1}, a_{i-2},\ldots, a_0$= 0.
        Step 1.1.2: Go to Step 4.

    Step 1.2: If $a_{i-1}$ =1 & $a_{i+1}$=0 then
        Step 1.2.1: Assign $a_{i-1}, a_{i-2},\ldots, a_0$ = 0.
        Step 1.2.2: Go to Step 4.

    Step 1.3: If $a_{i-1}$ =0 then
        Step 1.3.1: Assign $a_{i-1}, a_{i-2},\ldots, a_0$= 1.
        Step 1.3.2: If $a_{i+1}$=1 then
                        Assign $a_{i+1}$= 0.
                        Go to Step 4.
                    Else Go to Step 1.3.3.
        Step 1.3.3: Assign j=i+1.
                    Repeat for $a_j$!=1
                        assign $a_j$ =1
                        j=j+1.
                    Assign $a_j$=0.
                    Go to Step 4.

Step 2: If the 3rd bit of the amplitude value ($a_i$) is modified
        from 1 to 0 then
    Step 2.1: If $a_{i-1}$ =0 & $a_{i+1}$=0 then
```

```
            Step 2.1.1: Assign $a_{i-1}$, $a_{i-2}$, …, $a_0$= 1.
            Step 2.1.2: Go to Step 4.

        Step 2.2: If $a_{i-1}$ =0 & $a_{i+1}$=1 then
            Step 2.2.1: Assign $a_{i-1}$, $a_{i-2}$,….., $a_0$ = 1.
            Step 2.2.2: Go to Step 4.

        Step 2.3: If $a_{i-1}$ =1 then
            Step 2.3.1: Assign $a_{i-1}$, $a_{i-2}$,….., $a_0$= 0.
            Step 2.3.2: If $a_{i+1}$=0 then
                              Assign $a_{i+1}$= 1.
                              Go to Step 4.
                        Else goto step 2.3.3.
            Step 2.3.3: Assign j=i+1.
                        Repeat for $a_j$!=0
                                assign $a_j$=0
                                j=j+1.
                        Assign $a_j$=1.
                        Go to Step 4.
Step 3: if the 3rd LSB is same with the replaced bit then
        the amplitude value remain unchanged
        and Go to Step 4.
Step 4: return a.
```

Algorithm for Function STOR_LEN (STRLEN, Cover Audio)

```
/* Storing of the length of the message */

Step 1: Start
Step 2: Convert the value STRLEN into its 8 bit binary
        equivalent.
Step 3: Replace LSB of first 8 consecutive amplitudes
        with the 8 bits of STRLEN.
Step 4: End
```

Algorithm for Function M (L)

```
/* Finding the affected amplitude location */

Step 1: Start
Step 2: If L is equal to Len,
        Put P=L
        else
        Put P= (L+(L-1))*L
Step 3: Return P
Step 4: End
```

DECODING ALGORITHM OF HIDING DATA IN AN AUDIO FILE

```
Step 1: Start
Step 2: Take the Stego-audio file
Step 3: To obtain the length of the target text message call
        function RET_LEN (Stego Audio) and store
        the length in STRLEN.
Step 4: if STRLEN>8 then
          Set Len = STRLEN
        else
          Set Len = 8+STRLEN
        L=Len
Step 5: For 1 to STRLEN repeat the steps 6 to 14.
Step 6: Call the function M (L) to find out the amplitude
        location of the stego audio where the target text
        Massage is embedded.
Step 7: Convert the amplitude values into 8 bit binary.
Step 8: Cut the 3rd LSB and the 1st LSB of consecutive 3
        amplitude values according to the series.
Step 9: Concatenate the bits to obtain the 6 bit
        ($c_0c_1c_2c_3c_4c_5c_6$) equivalent of each character of
        string.
Step 10: If the values of ($c_0$, c1) are (0, 0) or (0, 1),
          put the 7th bit (MSB) as 1
         else if the values of ($c_0$,$c_1$) is (1,0) or (1,1)
          put 0 (refer table II ).
Step 11: Obtain the final 7 bit equivalent of each character
         of the string.
Step 12: Convert the 7 bit equivalent in the ASCII code of
         the individual characters.
Step 13: L=L+1.
Step 14: Concatenating them the target message is obtain.
Step 15: End
```

Algorithm for Function M (L)

```
/* Finding the affected amplitude location */

Step 1: Start
Step 2: If L is equal to Len,
          Put P=L
        else
          Put P= (L+(L-1))*L
Step 3: Return P
Step 4: End
```

Algorithm for Function RET_LEN (Stego Audio)

```
/* Recover the length on receiver side */

Step 1: Start
Step 2: Read first 8 amplitude value of the stego audio.
Step 3: Convert them in binary.
Step 4: Cut the LSB from each 8 bit binary equivalent of
        the amplitudes.
Step 5: Concatenate them to get the 8 bit binary equivalent
        of the length of the target string.
Step 6: convert this 8 bit binary into its corresponding
        decimal.
Step 7: Return
Step 8: End
```

VI. RESULT AND DISCUSSION

An English message text is written by using the alphabetic characters of the English language ((which are 26 letters (‘A’ … ‘Z’)) as well as numeric digits (which are 0 to 9). Some other special characters are also used to give the reader a proper understanding of the message. Here we consider only the uppercase alphabetic characters; numeric digits and some most commonly used special character for the better understanding of secret target message. The characters consider in this study are given in the following Table II.

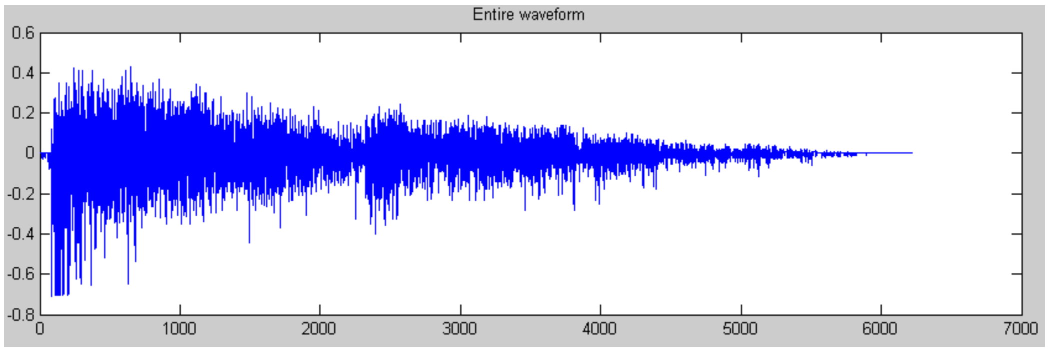
Table II

Character	**ASCII Code**	**Binary**				**Character**	**ASCII Code**	**Binary**				**Character**	**ASCII Code**	**Binary**			
		MSB	**R**	**G**	**B**			**MSB**	**R**	**G**	**B**			**MSB**	**R**	**G**	**B**
A	65	1	00	00	01	O	79	1	00	11	11	2	50	0	11	00	10
B	66	1	00	00	10	P	80	1	01	00	00	3	51	0	11	00	11
C	67	1	00	00	11	Q	81	1	01	00	01	4	52	0	11	01	00
D	68	1	00	01	00	R	82	1	01	00	10	5	53	0	11	01	01
E	69	1	00	01	01	S	83	1	01	00	11	6	54	0	11	01	10
F	70	1	00	01	10	T	84	1	01	01	00	7	55	0	11	01	11
G	71	1	00	01	11	U	85	1	01	01	01	8	56	0	11	10	00
H	72	1	00	10	00	V	86	1	01	01	10	9	57	0	11	10	01
I	73	1	00	10	01	W	87	1	01	01	11	.	46	0	10	11	10
J	74	1	00	10	10	X	88	1	01	10	00	,	44	0	10	11	00
K	75	1	00	10	11	Y	89	1	01	10	01	?	63	0	11	11	11
L	76	1	00	11	00	Z	90	1	01	10	10	!	33	0	10	00	01
M	77	1	00	11	01	0	48	0	11	00	00	blank	32	0	10	00	00
N	78	1	00	11	10	1	49	0	11	00	01	-	45	0	10	11	01

Before going into detailed discussion, at the sender side we choose the information of the target message and a cover audio file under which the target message is to be hidden.

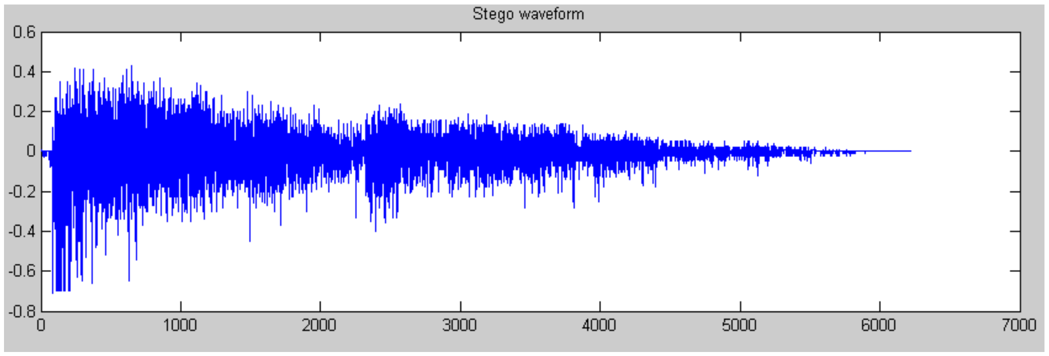
Test Case 1:

Cover Audio: Glass.wav



Target message: AUDIO

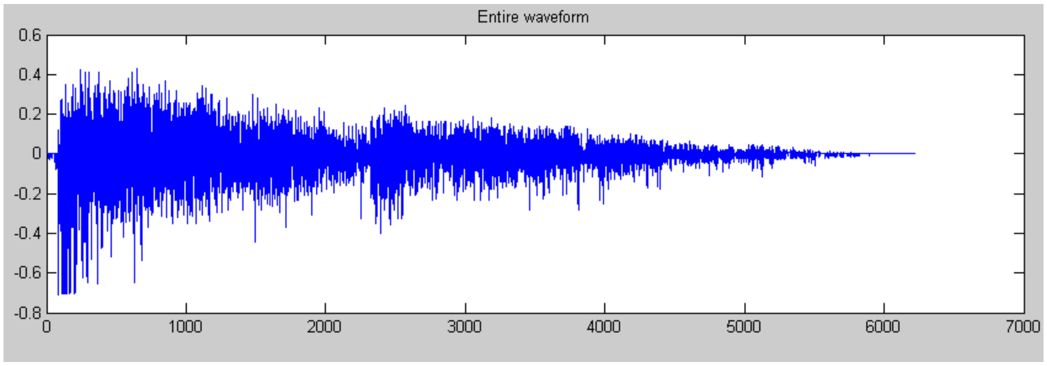
Stego Audio: GA.wav



Retrieved message: AUDIO

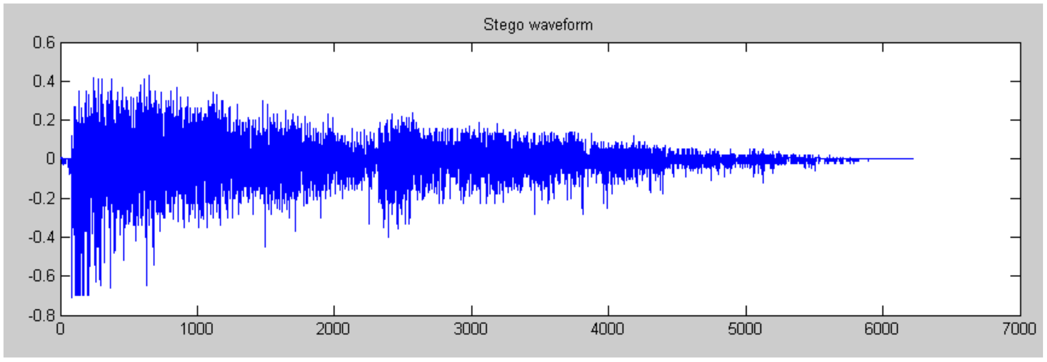
Test Case 2:

Cover Audio: Glass.wav



Target message: STEGANOGRAPHY

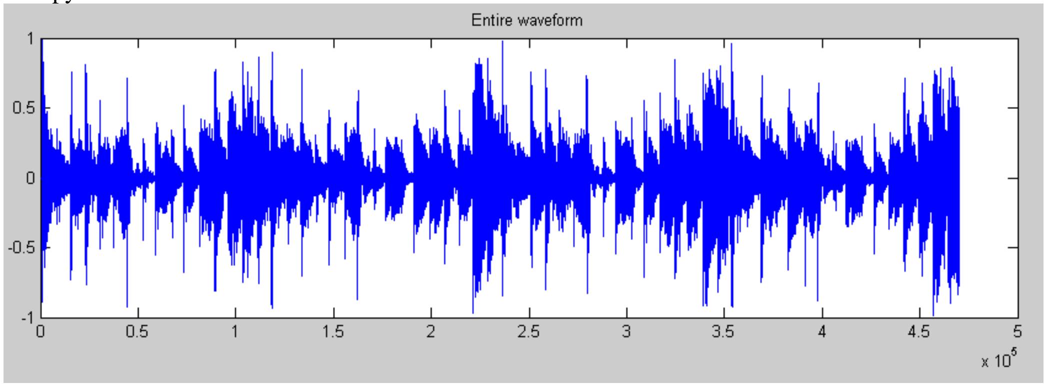
Stego Audio: GS.wav



Retrieved message: STEGANOGRAPHY

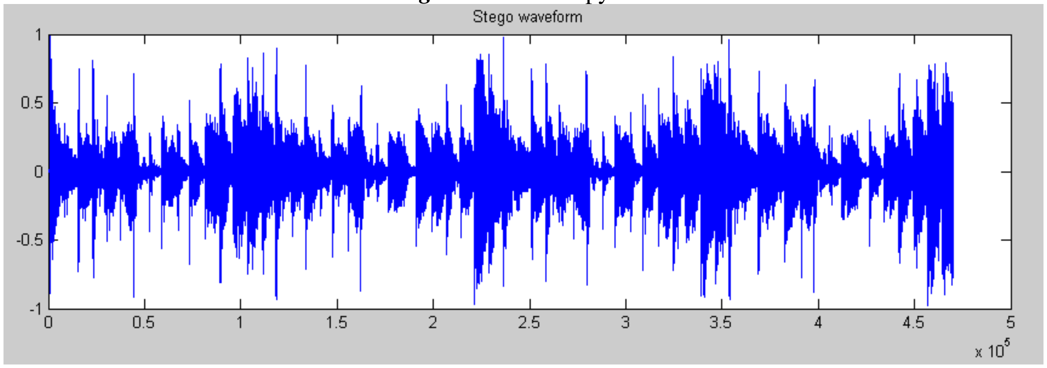
Test Case 3:

Cover Audio: LoopyMusic.Wav



Target Message: THIS IS A STEGO AUDIO

Stego Audio: Gloopy.wav



Extracted Message: THIS IS A STEGO AUDIO

For detailed discussion we consider the test case 1. In test case 1
Target message: audio
Cover audio: Glass.wav
Before embedding process we need to convert the target message into upper case. Then we store the length of the message in the first 8 amplitudes by replacing the LSB of each it. Then we start hiding data using our proposed method.
Now, we start our work with the target string "AUDIO". We encode this string according to the proposed method as shown in Table III.

TABLE III

Charact-er of Target String	ASCII value of correspon-ding characters	MSB of the 7 bit ASCII	Binary 6 bit equivalent of the character	Affected amplitude number	Original amplitude value in decimal	Original amplitude value in 8 bit binary	Modified amplitude value in 8 bit binary	Modified amplitude value in decimal
A	65	1	000001	13 325 378	0 0 26	00000000 00000000 00011010	00000000 00000000 00011011	0 0 27
U	85	1	010101	435 496 561	0 23 (-)23	00000000 00010111 00010111	00000001 00011001 00011001	1 25 (-)25
D	68	1	000100	630 703 780	31 (-)15 (-)16	00011111 00001111 00010000	00100000 00010001 00010000	32 (-)17 (-)16
I	73	1	001001	861 946 1035	(-)23 (-)14 (-)1	00010111 00001110 00000001	00011000 00001110 00000001	(-)24 (-)14 (-)1
O	79	1	001111	1128 1225 1326	15 (-)15 (-)9	00001111 00001111 00001001	00010000 00001111 00000111	16 (-)15 (-)7

In the case when the original and hidden bit are different and ith LSB layer is used for embedding the error caused by embedding is 2^i-1 quantization steps (QS). The embedding error is positive if the original bit was 0 and watermark bit is 1 and vice versa. Our proposed LSB algorithm causes minimal embedding distortion of the host audio. We can see from column 6 & 7 that even after changing the 3rd LSB, the value is not changed significantly. In fact the changes are so minimal that they cannot affect the HAS. This is because of the fact that we have manipulated the binary string in such a way that the change in the 3rd LSB is compensated to a great extent by some other bits. We have used a unique method to hide data bits in the 3rd LSB and in the 1st LSB as explained above. A change in the 3rd bit means that there can be a maximum change of 4. But we can clearly see from Table VI that the maximum change in the amplitude values is 2 and after adjustment when we replace 1st LSB the maximum change becomes 3 for some cases. But here we can hide two bits rather than one in single amplitude. So, this method is quite efficient. Also, the security as well as robustness has increased a lot because the intruders will never have any idea that data can be hidden in the 3rd LSB just because any change in the 3rd bit makes a great change. Generally, bits are hidden either in the 1st position or sometimes in both the 1st LSB and the 2nd LSB. But our method is very unique and advantageous in this aspect.

During the extraction of secret target message at receiver site first the length of the hidden message is retrieved from the LSB of first 8 amplitudes. Now According to the defined series we find out the amplitude position where the data bits are embedded. The whole extraction process of data at the receiver side is depicted in Table IV.

TABLE IV

Affected amplitude number	Amplit-ude value in decimal	Amplitud-e value in 8 bit binary	Extract 3rd LSB and 1st LSB	Concat-enated 6 bit	7th bit(MSB) of the character	Concate -nated 7 bit	Correspon -ding ASCII	Correspon -ding character of target string
13 325 378	0 0 27	00000000 00000000 00011011	00 00 01	00 00 01	1	1 00 00 01	65	A
435 496 561	1 25 (-)25	00000001 00011001 00011001	01 01 01	01 01 01	1	1 01 01 01	85	U
630 703 780	32 (-)17 (-)16	00100000 00010001 00010000	00 01 00	00 01 00	1	1 00 01 00	68	D
861 946 1035	(-)24 (-)14 (-)1	00011000 00001110 00000001	00 10 01	00 10 01	1	1 00 10 01	73	I

1128 1225 1326	16 (-)15 (-)7	00010000 00001111 00000111	00 11 11	00 11 11	1	1 00 11 11	79	O

From TABLE VII, we can see that after obtaining the 6 bit equivalent of each and every character of the target string we concatenate the 7th bit (MSB) based on the 6th and the 5th bit of 6 bit equivalent (from the LSB side), 0 if they are 11 or 10, 1 if they are 00 or 01. Convert the 7 bit to ASCII from which the target text can be retrieved.

From table VI we can see that the changes in amplitude values are so minimal that it cannot affect in human eyes. If we follow the column "Affected amplitude number" from table VI & VII we can send a large message through a 3 to 4 minute audio clip. Here to store target message of size 21 we require an audio clip with minimum 2000 samples. So when we send this type of stego image through internet it takes lesser bandwidth.

Here we consider only the uppercase alphabetic character for interpretation of characters from the stego message at the receiver side. Otherwise according to our hidden process some of the lower case letters in 6 bit match with 6 bits of some mostly used special characters that we consider in this study. But in this manner (i.e. only send 6 bits) we are able to send lesser bits i.e. compressed target message to the receiver thus increasing the efficiency of the method. The receiver finally obtains the target string by merging all the characters.

CONCLUSION

Steganography can be an effective means that enables concealed data to be transferred inside of seemingly innocuous carrier files. Generally speaking, steganography brings science to the art of hiding information. The purpose of steganography is to convey a message inside of a conduit of misrepresentation such that the existence of the message is both hidden and difficult to recover when discovered. Many different techniques exist and continue to be developed for hiding information for secure transaction. But we can not find out a technique which gives ultimate security. As well as detection can never give a guarantee of finding all hidden information. Even then, perfect steganography, where the secret key will merely point out parts of a cover source which form the message, will pass undetected, because the cover source contains no information about the secret message at all.

The main goal of our research work was embedding of text into audio as a case of steganography. The two primary criteria for successful steganography are that the stego signal resulting from embedding is perceptually indistinguishable from the host audio signal, and the embedded message is recovered correctly at the receiver using some easiest technique are successfully followed by our method according to test.

In this paper we only work with .wav file. In future we try to apply it on other audio file format. Here we have hide a string in an audio file. We wished if we could hide an audio inside another audio. In other words rather than having a target string we could have a target audio clip. In this paper we follow a series for selecting amplitudes for hiding information but our aim is to place the target message within a particular threshold. We could also have applied the concept of encryption and decryption rather than encoding and decoding. Steganography and cryptography can be used together to make it more secure and reliable. However future scope appears end less.

REFERENCES

[1] Chang C. C., Chen T. S. and Hsia H. S.:"An Effective Image Steganographic Scheme Based on Wavelet Transform and Pattern- Based Modification", IEEE Proceedings of the 2003 International Conference on Computer Networks and Mobile Computing, 2003.

[2] Chen and Womell G.W.: "Quantization index modulation: a class of provably good methods for digital watermarking and information embedding", IEEE Transactions on Information Theory, Vol. 47, No. 4, pp. 1423-1443, May 2001.

[3] Chen B.: "Design and analysis of digital watermarking, information embedding, and data hiding systems," Ph.D. dissertation, MIT, Cambridge, MA, June 2000.

[4] Matsuoka H.: "Spread Spectrum Audio Steganography using Sub – band Phase Shifting", Proceedings of the 2006 International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIHMSP' 06), IEEE, 2006

[5] Agaian S.S., Akopian D., Caglayan O., D'Souza S. A., "Lossless Adaptive Digital Audio Steganography," In Proc. IEEE Int. Conf. Signals, Systems and Computers, pp. 903-906, November 2005.

[6] Gopalan K., "Audio steganography using bit modification", Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing, Vol. 2, pp. 421-424, April 2003.

[7] Mohammad P., Ahmed D., "LSB-based Audio Steganography Method Based on Lifting Wavelet Transform", International Symposium on Signal Processing and Information Technology, IEEE, 2007.

[8] Basu P. N., Bhowmik T.: "On Embedding of Text in Audio – A case of Steganography", International Conference on Recent Trends in Information, Telecommunication and Computing, 2010

[9] Parthasarathy C., Srivatsa S.K.: "Increased Robustness of LSB Audio Steganography by Reduced Distortion LSB Coding" Journal of Theoretical and Applied Information Technology. Vol 7. No 1. (pp 080 - 086), 2005 - 2009

[10] Sridevi R, Dr. Damodaram A, Dr. Narasimham Svl.: "Efficient Method of Audio Steganography by Modified LSB Algorithm and Strong Encryption Key with Enhanced Security" Journal of Theoretical and Applied Information Technology, 2005 – 2009.

[11] Cvejic N., Seppaanen T.: “Seppaanen Increasing Robustness of LSB Audio Steganography by Reduced Distortion LSB Coding”. Journal of Universal Computer Science, vol. 11, no. 1 (2005), 56-65.

[12] Audio Engineering Society E-Library - Steganographic Approach to Copyright Protection of Audio; Preprint Number: 7067 Convention: 122 (May 2007).

[13] [14] Johnson N.F and Katzenbeisser S: “A survey of steganographic techniques”. Information Hiding, Artech House,pp.43-78,2000.

[15] J. Foote, A Similarity Measure for Automatic Audio Classification, *Proc. AAAI 1997 Spring Symp. on Intelligent Integration and Use of Text, Image, Video, and Audio Corpora,* Stanford, CA, 1997.

[16] J. J. Burred and A. Lerch, Hierarchical Automatic Audio Signal Classification, J. Audio Eng. Soc, Vol. 52, pp. 724-739, July/August 2004.

[17] K. Gopalan, “Audio steganography using bit modification”, Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing, Vol. 2, pp. 421-424, April 2003.