ABSTRACT

Audio steganography deals with a method to hide a secret message in an audio file. Also, Audio steganography can be used for secret watermarking or concealing ownership or copyright information in the audio that can be verified later to justify ownership right. A direct Least Significant Bit (LSB) substitution method is one of the most simple and popular techniques used for audio steganography; however, the drawback of this technique is that it can cause hissing sound when the stego audio is played. On the other hand, Human Audio System (HAS) is highly sensitive that can detect or discriminate subtle changes in an audio. In order to improve robustness against hissing noise as well as to avoid detection by a human being, we present a threshold-based scheme for an audio steganography that can selectively embed bits in audio samples based on their magnitudes. Specifically, the scheme can be used to embed zero or many secret bits in an audio sample depending on the relative strength of the audio sample as the sample is measured against a scale of multi-level threshold value. The scheme selects threshold values by analyzing the audio file for a given secret message so that the overall error can be as small as possible while fully utilizing the capacity of the audio file.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. BACKGROUND AND RATIONALE

1.1 Audio Steganography

Steganography is an art of hiding a secret message in another message which everybody does not know about presence of the secret message except the intended receiver. The message used to hide the secret message is called a host message or cover. The secret message can be a text, image, audio, or video. The combination of host message and secret message is called stego message. Image, audio and video files are considered as excellent cover files due to presence of redundancy [1]. However, audio steganography is considered more difficult than image steganography and video steganography since the Human Audio System (HAS) is more sensitive than Human Visual System (HVS) [1].

There are many factors that can determine success (or failure) of audio steganography. Human memory plays an important role in detection of existence of a secret message in a stego audio. Hence, the cover audio should be an unknown audio because a slight change in a known audio can raise suspicion due to a perceptual difference sensed between the original and the modified audio.

In order to embed the secret message successfully, the adopted technique should be capable against HAS. Any audio steganography should satisfy three requirements: capacity, transparency, and robustness [1]. Capacity means the amount of secret data is hidden in the stego message while transparence deals with how well the secret message is embedded in the stego message. Robustness is the ability of embedded the secret message to against attacks. It is difficult to retrieve the secret message.

In this work, we propose an audio steganographic scheme that considers of HAS carefully so that the applications does not have to rely heavily on whether known or unknown audio is being used as cover. This scheme uses selected audio samples from covered audio to embedded secret message bits. Selection of audio samples is based on a multi-threshold scheme. Also, capability is a concern in this paper. Depending on relative change, there are different capabilities. In the next section, Least Significant Bit substitution will be discussed.

1.2 Least Significant Bit Substitution

The LSB substitution is a simple and fast method in which provides high capacity for data [1]. In this method, each bit of the secret message is embedded in the LSB of cover audio in a specific way. The length of the secret message should be smaller than the total number of samples in a cover audio. Figure 1.1 shows how the secret message "HI" is embedded in 16-bit sample using the LSB modification method.

There exist many steganographic schemes for hiding secret messages in audio bits. In [2] and [3], Cvejic and Seppanen proposed a scheme which increases the depth of the embedding layer from fourth LSB layer to the sixth LSB layer. In this scheme, only bits at the 6^{th} position of each 16 bits sample are replaced with bits of the secret message. In [4] and [5], the authors proposed the schemes which embed secret message bits in the wavelet domain. The encrypted data is embedded into the wavelet coefficients of the host audio signal. In [6], Matsuoka proposed sub-band phase shifting as the method of processing the original audio signal for embedding a secret message.

Sampled Audio Stream (16-bit)	"HI" in binary	Audio Stream with message encoded
1 1 0 1 0 1 1 0 1 0 1 0 1 0 1 0	0	1 1 0 1 0 1 1 0 1 0 1 0 1 0 **0**
0 1 0 1 1 0 1 0 1 0 1 0 1 0 1 1	1	0 1 0 1 1 0 1 0 1 0 1 0 1 0 **1**
1 0 0 0 1 1 1 0 0 1 0 1 0 1 1 1	0	1 0 0 0 1 1 1 0 0 1 0 1 0 1 1 **0**
0 0 0 1 1 0 1 0 0 1 1 0 1 0 1 1	0	0 0 0 1 1 0 1 0 0 1 1 0 1 0 1 **0**
0 0 1 0 1 1 1 1 0 1 0 1 1 1 1 0	1	0 0 1 0 1 1 1 1 0 1 0 1 1 1 1 **1**
0 1 0 1 0 1 0 1 0 1 1 0 1 0 0 0	0	0 1 0 1 0 1 0 1 0 1 1 0 1 0 0 **0**
1 1 0 0 0 1 1 0 1 1 0 0 1 1 0 0	0	1 1 0 0 0 1 1 0 1 1 0 0 1 1 0 **0**
1 1 1 1 0 0 0 1 1 0 1 0 1 0 1 0	0	1 1 1 1 0 0 0 1 1 0 1 0 1 0 1 **0**
0 1 1 0 0 1 1 0 1 0 0 0 0 1 1 0	0	0 1 1 0 0 1 1 0 1 0 0 0 0 1 1 **0**
1 1 1 0 1 0 1 0 1 0 1 1 1 1 0 0	1	1 1 1 0 1 0 1 0 1 0 1 1 1 1 0 **1**
0 0 0 1 1 1 0 1 0 0 1 1 0 0 1 0	0	0 0 0 1 1 1 0 1 0 0 1 1 0 0 1 **0**
0 1 0 1 1 0 1 1 0 0 1 1 0 0 0 1	0	0 1 0 1 1 0 1 1 0 0 1 1 0 0 0 **0**
1 1 0 0 1 0 1 0 1 1 0 1 0 1 0 0	1	1 1 0 0 1 0 1 0 1 1 0 1 0 1 0 **1**
1 1 0 1 1 0 1 0 1 0 0 1 0 1 0 0	0	1 1 0 1 1 0 1 0 1 0 0 1 0 1 0 **0**
1 0 1 1 0 0 1 1 0 0 1 0 1 0 1 1	0	1 0 1 1 0 0 1 1 0 0 1 0 1 0 1 **0**
0 0 1 0 1 1 0 0 1 0 1 0 1 0 1 0	1	0 0 1 0 1 1 0 0 1 0 1 0 1 0 1 **1**
↑ LSB column		

Figure 1.1 LSB Substitutions

Our scheme is based on bit substitution method in time-domain. We do not embed a bit or bits in a sample unless its amplitude is high enough based on some threshold criteria.

1.3 Echo Hiding

Echo hiding method is another technique for audio steganography. In this method, a secret message is embedded in a cover audio file by introducing a short echo into the distinct signal. In order to hide the secret message, the echo hiding method uses three parameters of the echo signal: amplitude, decay rate and offset [7]. Since a delay is up to ms between the cover audio and the echo, it is difficult to distinguish. However, the disadvantage of echo hiding is that it allows embed a short message because of its low capacity [8]. The author in [8] stated that this technique is generally used for watermarking.

Phase Coding

The basic idea of phase coding technique is to split a cover audio into small blocks whose lengths equal a size of a secret message. The secret message only is embedded into the phase spectrum of the first block. In [8], the authors pointed that phase coding technique tolerates signal distortion better than other techniques. However, the drawback of phase coding is its low embedding capacity that allows only the first block is used to embed the secret message.

2. NARRATIVE

This chapter describes the theory and algorithms behind how threshold values are selected, the capacity of a given audio is calculated, the embedding process is carried out to embed a secret message in the audio file and the retrieving process is accomplish to retrieve the secret message.

2.1 Scope of The Project

The main purpose of this project is to study a new scheme and apply it to audio steganography. The scheme can embed zero or many secret bits to audio samples based on multi threshold values. Specifically, this projects aims to create a system so that users can hide a secret message to an audio file.

2.2 Threshold Values

This section discusses a multi-threshold based scheme that allows selecting only adequately large audio sample from cover audio for embedding secret message bits. In this section, threshold condition, recoverability and selection of threshold value are described.

2.2.1 Threshold Condition

Let C be an array of audio samples of the cover audio and S be an array of audio samples of the stego audio which is generated after embedding the given secret message. Since S[i] is generated by embedding secret bits to C[i], the difference between C[i] and S[i] can be defined as an error $e[i] = |C[i] - S[i]|$ where i is $(0 \leq i \leq \text{number of samples})$. This error e[i] depends on how many bits are embedded in the audio sample C[i]. If we embed k secret bits in LSB positions of C[i], the error e[i] can be as large as $2^k - 1$. For

example, if we embed 1111 as four secret bits in the original sample C[i] which had 0000 as least significant bits, the error would be 15.

Our goal is to embed fewer bits in audio samples with smaller values and more bits in larger audio samples. The reason is that if we embed too many secret bits to audio samples having small values, it can cause noticeable noise thus foiling the purpose of audio steganography. For example, embedding secret bits in a long silent segment of a cover audio sample can cause hissing sound. Thus, it is better to avoid or limit embedding secret bits to the cover audio sample with small values. By carefully selecting a threshold condition, one can limit the number of secret bits that can be embedded in an audio sample so as to reduce noise in the stego audio. Also, characteristics of the cover audio document as well as distribution of secret bits throughout the cover audio can determine the quality of the stego audio. It will be hard to detect if secret bits are sparsely distributed in a cover audio. By selecting high threshold values one can achieve a desired distribution of secret bits whether sparse or dense.

We assume that cover audio samples in C are classified into (p + 1) categories to embed 0, 1, 2, 3, …, p secret bits depending on their magnitude and/or some error constraint. Accordingly, let T[1], T[2], T[3], …, T[p] represent threshold values, sorted in ascending order, that will be used to embed 1, 2, 3, . . ., p bits. Specifically, in order to embed k bits in an audio sample C[i], the following condition must be satisfied:

$$T[k] \leq C[i] < T[k + 1]$$

It is to be noted that if C[i] < T[1], no bits can be embedded in the audio sample. On the other hand, if C[i] ≥ T[p], we assume that at most p bits can be embedded in C[i].

Hence p is maximum numbers of bits that can be allowed to be embedded in a cover audio.

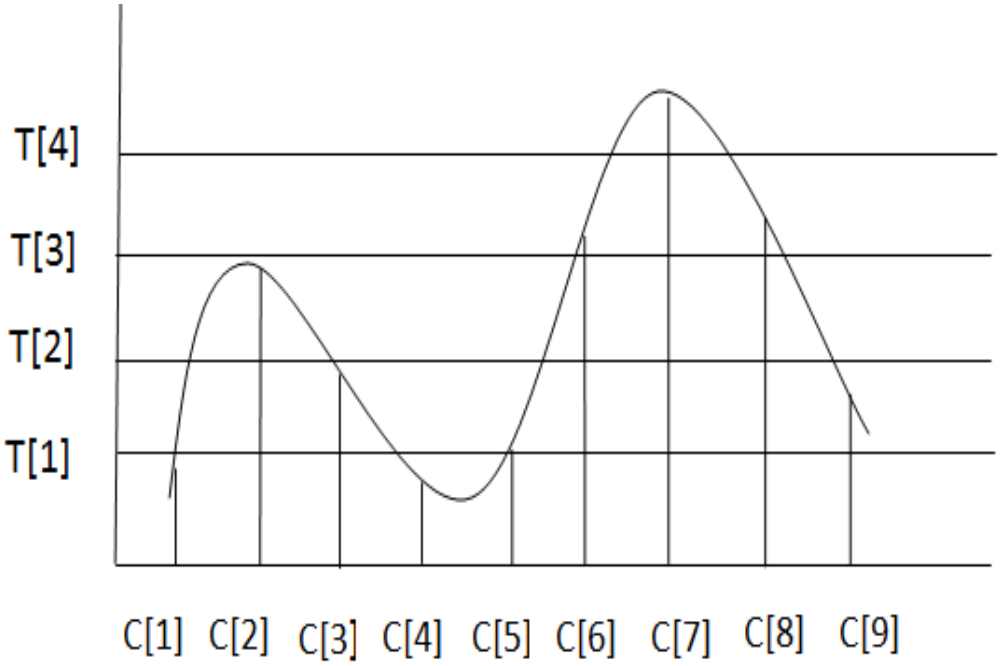

Figure 2.1 Threshold Values

Figure 2.1 shows how audio samples can be categorized into five distinct groups based on threshold values T[1], T[2], T[3], an T[4]. Correspondingly, Table 2.1 summarizes the results of grouping based on the threshold values. No bits can be embedded in any sample of group 1, since its value is less than threshold value T[1]. However, a sample in group 2, group 3, group 4, or group 5 can be used to embed 1, 2, 3, or 4 secret bits.

Table 2.1 Audio Samples

Group 1	Group 2	Group 3	Group 4	Group 5
C[1], C[4], C[5]	C[3], C[9]	C[2]	C[6], C[8]	C[9]

Selection of threshold conditions is extremely important because it affects the quality of stego audio. Besides selecting the threshold values, recoverability of secret bits from stego audio should be considered as well. During recovery, we need to determine how many bits have been embedded in a stego sample and accordingly extract the correct number of bits from the sample. The following section describes the recoverability condition so that secret message bits could recovery from stego audio successfully.

2.2.2 Recoverability

Given a stego sample S[i], we need to know how many bits have been embedded in it. However, after embedding secret message bits, the stego sample S[i] can be larger or smaller than the cover audio sample C[i]. This can lead to the stego audio sample S[i] to be smaller than T[k] or greater than or equal to T[k+1]. For example, let us suppose the value of the cover audio sample C[i] is 11010 (26 in decimal) and the values of the threshold values T[1] and T[2] chosen are 24 and 27 respectively. After embedding one as a bit to the audio sample C[i], the value of audio sample S[i] will be 11011 (27 in decimal). This can lead to a wrong result since the threshold condition for retrieving a single bit will be violated.

Thus, it is difficult to identify how many bits have been embedded in a modified audio sample S[i]. It is to be noted that the modified audio sample S[i] is obtained by embedding k bits in a cover audio sample C[i]. By comparing S[i] with the threshold values during the recovery process, we can identify that S[i] has only k secret bits. T[k] is the minimum threshold value of an audio sample for embedding k bits in an audio sample while T[k + 1] is the minimum threshold value of an audio sample for embedding(k + 1) bits. Hence, the largest stego sample S[i] has to be less than T[k+1] and the smallest stego

sample S[i] must not be less than T[k]. Thus, the largest audio sample obtained after embedding k bits should be less than T[k +1] and the smallest audio sample after embedding k bits should be greater or equal to T[k].

$$\text{Max } S[i] < T[k+1] \text{ and Min } S[i] \geq T[k] \quad (1)$$

If S[i] is greater than T[k+1], there is an erroneous extraction of (k + 1) or more bits (k - 1) or less if S[i] is less than the threshold value T[k]. It is necessary to check condition during recovery to prevent errors. In order to ensure that S[i] contains only k embedded bits, S[i] must satisfy:

$$T[k] \leq S[i] < T[k + 1] \quad (2)$$

In order satisfy condition (2), C[i] must be checked during substitution of its k LSBs with secret message bits. Based on shifting right k position, the minimum value for S[i] occurs when:

$$S[i] = \lfloor C[i]/2^k \rfloor * 2^k$$

It means when all substituted k least significant bits in C[i] are zero. Besides, the maximum value for S[i] occurs when:

$$S[i] = \lfloor C[i]/2^k \rfloor * 2^k + 2^k + 1$$

This happens when all substituted k LSBs in C[i] are one. For example, we have the audio sample C[i] = 000110110 (54 in decimal). After shifting right by 2 positions, we have 000001101 ($\lfloor C[i]/2^k \rfloor = \lfloor 54/2^2 \rfloor = 13$). Since sample C[i] embed 2 bits, the minimum value of S[i] is 00110100 ($\lfloor C[i]/2^k \rfloor * 2^k = \lfloor 54/2^2 \rfloor * 2^2 = 52$) and the maximum value of S[i] is 00110111($\lfloor C[i]/2^k \rfloor * 2^k + 2^k + 1 = \lfloor 54/2^2 \rfloor * 2^2 + 2^k + 1 = 57$).

Before substitution of k least significant bits in C[i], it is necessary to check both possible minimum and maximum values of S[i] are bounded by condition (2). Hence, the following condition needs to be satisfied:

$$T[k] \leq \lfloor C[i]/2^k \rfloor * 2^k \text{ and } \lfloor C[i]/2^k \rfloor * 2^k + 2^k + 1 < T[k+1] \quad (3)$$

Alternatively, it is also possible to choose a threshold value T[k] for all k = 1 to p as a binary number with all k LSBs as zero and then check condition (1) for all situations. In this case, when condition (1) is satisfied, condition (2) will be satisfied. This works because $T[k] \leq C[i]$ implies $T[k] \leq \lfloor C[i]/2^k \rfloor * 2^k$ and $C[i] < T[k+1]$ implies $\lfloor C[i]/2^k \rfloor * 2^k + 2^k + 1 < T[k+1]$ as well. Though convenient, however, choosing such values for T[k] and T[k+1] will widen the range between two successive thresholds drastically as it cause $T[k+1] = 2^1 T[k] = 2^2 T[k-1] \ldots = 2^k T[1]$. By setting a convenient value for T[1], any other threshold value such as T[k] can be obtained just by shifting T[1] to the left by (k - 1) bits.

It is to be noted that if T[k] and T[k+1] are arbitrarily chosen to close to each other such that $T[k+1] - T[k] \leq 2^k - 1$ then there will not be any audio sample satisfying condition; therefore, there will not be any sample to be embedded with k secret message bits.

2.2.3 Selection of Threshold Values

Selecting threshold values can be very difficult since we would like to maximize the capacity of the cover audio to embed as many secret bits as possible. In addition, we would like to minimize distortion in the resultant audio as much as possible to avoid detection. Hence, in order to find an optimum of set of threshold values, we attempt to limit the change in each specific sample as small as possible.

First, let $\varepsilon_{k,i}$ represent the relative change in a sample C[i] after embedding k secret bits in it. Let $\varepsilon_{k,i}$ be defined as:

$$\varepsilon_{k,i} = \frac{|C[i] - S[i]|}{C[i]} \quad (4)$$

The relative change $\varepsilon_{k,i}$ becomes maximum when $|C[i] - S[i]|$ is maximum and C[i] is small as possible meeting the threshold condition:

$$T[k] \leq C[i] < T[k+1]$$

As we stated above, the largest change or error in an audio sample after embedding k bits can be as large as (2^k -1). Thus, $|C[i] - S[i]|$ can be a maximum value when:

$$|C[i] - S[i]| = 2^k - 1 \quad (5)$$

From $T[k] \leq \lfloor C[i]/2^k \rfloor * 2^k$ in the equation (3), it is can be expressed as

$$C[i] \geq \lceil T[k]/2^k \rceil * 2^k$$

Hence, the smallest audio sample C[i] that does not violate the recoverability condition is

$$C[i] = \lceil T[k]/2^k \rceil * 2^k \quad (6)$$

Let ε_k represent the maximum of all relative change that occur in all C[i] in the range. According to equations (4), (5), and (6), the largest relative change after embedding k secret message bit can be given by

$$\varepsilon_k = \frac{2^k - 1}{\lceil T[k]/2^k \rceil * 2^k} \quad (7)$$

Equation (7) can be used to select threshold values T[1], T[2],…, T[p] in such a way that no sample-wise relative change does not exceed some given limit. Let δ represent such limit, we have $\varepsilon_k \leq \delta$ for all k = 1, 2, …, p.

$$\varepsilon_k = \frac{2^k - 1}{\lceil T[k]/2^k \rceil * 2^k} \leq \delta \qquad (8)$$

From equation (8), for a given limit δ, we obtain the lowest possible value of T[k] as

$$T[k] = 2^k * \lceil (2^k - 1)/2^k \delta \rceil \qquad (9)$$

Table 2.2 Sample Threshold Values for $0.0001 \leq \delta \leq 0.0005$

Allowed Maximum Relative Change, δ	T[1]	T[2]	T[3]	T[4]	T[5]
0.0001	10000	30000	70000	150000	310016
0.0002	5000	15000	35000	75008	155008
0.0003	3334	10000	23336	50000	103360
0.0004	2500	7500	17504	37504	77504
0.0005	2000	6000	14000	30000	62016

Table 2.1 shows some sample threshold values for $0.0001 \leq \delta \leq 0.0005$ respectively. In the table we show first five threshold values T[1], T[2], T[3], T[4], and T[5] for different choices of δ. For example, we want to restrict any relative change in samples by $\varepsilon_k \leq 0.0001$ for embedding k bits. The corresponding threshold values will be: 10000, 30000, 70000, 150000, and 310008 for embedding 1, 2, 3, 4, and 5 bits respectively.

$T[1] = 2^1 * \lceil (2^1 - 1)/2^1 * 0.00001 \rceil = 10000$

$T[2] = 2^2 * \lceil (2^2 - 1)/2^2 * 0.00001 \rceil = 30000$

$T[3] = 2^3 * \lceil (2^3 - 1)/2^{13} * 0.00001 \rceil = 70000$

$T[4] = 2^4 * \lceil (2^4 - 1)/2^4 * 0.00001 \rceil = 150000$

$T[5] = 2^5 * \lceil (2^5 - 1)/2^5 * 0.00001 \rceil = 310016$

Selecting threshold values in this way will ultimately limit error or change at each sample C[i] because of the constraint $T[k] \leq C[i] < T[k+1]$. It guarantees that $| (C[i] - S[i] | / C[i] \leq \delta$ for all values of i = 1, 2, …, n will be maintained. It is expected that limiting changes in each individual audio sample will ultimately limit overall distortion in the stego audio. In order to avoid suspicion and detection, the value for δ should be adjusted depending on the type and quality of the cover audio. The following section describes how to find the suitable sample-wise error δ that the capacity embeds secret data in cover audio as much as possible.

2.3 Capacity

We define capacity as the maximum number of bits that can be embedded in given audio file by maintaining some error condition. This section describes how to select a sample-wise error which can be as small as possible while fully utilizing the capacity of the audio file to embed a given secret message file. The selection of a sample-wise error is one of the challenges of the scheme. If we choose a small sample-wise error δ, it will introduce less noise in the resultant stego audio; however, it may not lead to enough capacity to hold all the bits of a given secret message in the cover audio. On the hand, if we choose a large value for δ, it can introduce too much noise in the stego audio. We need to determine the minimum value of δ that will allow us to embed the secret message bits in the given audio so that noise will remain as least as possible.

Typically to compute the capacity of a cover audio, we need to check each audio sample with the given threshold values and accordingly determine how many bits can be embedded in the sample. Let N[i] be the number of bits that can be embedded in an audio

sample C[i] . Evidently the capacity of the cover audio C is $\sum_{i=0}^{n} N[i]$ where n is the number of samples in C. In order determine the optimum value for δ, we will need to compute the capacity under various threshold conditions before converging to the optimal solution. The direct summation approach can be computationally inefficient since we will need to read each sample every time we compute the capacity. Instead we propose an algorithm that can compute the capacity efficiently as it reads the cover audio file only once. The algorithm first creates an auxiliary array as explained in the following.

We assume that the cover audio has n samples in the range from 0 to k where k can be as large as 32768 in magnitudes. Let Count[0..k] be an auxiliary array in which an element Count[i] represents the number of samples in the cover audio C[] that have the same value which is equal to i. The following algorithm shows how to find sample-wise counts for the samples in the cover audio C[].

SAMPLE-WISE COUNTING ALGORITHM:

1. for 1 ← 0 to k

   do Count[i] ← 0

2. for j ← 1 to n

   do Count[C[j]] ← Count[C[j]] +1

3. for i ←1 to k

   do Count[i] ← Count[i] + Count[i -1]

By using the Count[] array and given threshold values, we can calculate the capacity of the cover audio C[] as described in the following.

FAST COMPUTATION OF CAPACITY:

Capacity(p)

1. capacity ← 0
2. for i ← 1 to p
3. capacity ← capacity + i * (Count[T[i + 1]] - Count[T[i] - 1])
4. capacity ←capacity + p * (Count[k] - Count[T[p] - 1])
5. return capacity

In order to find the sample-wise as small as possible while fully utilizing the capacity of the audio file, we propose the following algorithm to determine the sample-wise error.

FINDCapacity (n)

1 Min_error ← 1/Max (C)

2 Max_error ← 1

3 do

4 Mid_error ← (Min_error + Max_error)/2

5 n ← findthreshold(Mid_error);

6 capacity ←Capacity(n)

7 If (capacity < numberof_message_bits)

8 Min_error ← Mid_error

9 If capacity > numberof_message_bits

10 Max_error ← Mid_error

11 **While** ((capacity = numbeof_message_bit) || (|Max_error – Min_error| ≤ 1.0e-6)

We assume that the maximum sample-wise error is one, called Max_error, while the minimum sample-wise error is one per maximum amplitude of the cover audio, Min_error = 1/ max(C). Accordingly, the middle sample-wise error, named Mid_error, is (Max_error + Min_error)/2.

2.4 Threshold Value and Capacity Example

Let us assume that the array C[] stores the samples of the cover audio whose amplitudes are in the range from 0 to 9, C[0, 3, 8, 8, 5, 6, 9, 1, 1, 2, 9,…]. Also, let us assume that there is a secret message which includes 30 bits only. Accordingly, sample-wise counts can be shown as in Table 2.3.

Table 2.3 The Array Count[]

	0	1	2	3	4	5	6	7	8	9
Count (Step1)	0	0	0	0	0	0	0	0	0	0
Count (Step2)	5	2	9	6	3	4	1	0	10	20
Count (Step3)	5	7	16	22	25	29	30	30	40	60

To calculate the capacity of C[] in this case, let us assume that T[1] =1, T[2] = 8, and T[3] = 16. These threshold values are derived by following equation (9). Next, we assume that Min_err = 1/9 and Max_err = 1. Accordingly, we calculate Mid_err = (1/9 + 1)/2 = 0.556. After that, based on threshold values, the number of samples that can be used to embed a single bit in each of them are 30 - 7 = 23 and the number of samples that can be used to embed two bits in each sample are 60 - 30 = 30. Hence, the total capacity

is 23 + 30 = 53 bits. This value is greater than the secret message bits; therefore, we keep calculating the Mid_err by setting Max_err = Mid_err. Doing the same above steps, the threshold values and the capacity are calculated as shown in the following table.

Table 2.4 Threshold Values and Capacity

Itera _tion	Min _err	Max _err	Mid _err	T[1]	T[2]	T[3]	Capacity to insert			
							1bit	2bits	3bits	Total
1st	1/9	1	0.556	2	8	16	23	30	0	53
2nd	1/9	0.556	0.334	4	12	24	38	0	0	38
3rd	1/9	0.334	0.222	6	16	32	30	0	0	30

2.5 Embedding Process

From the given input files for a cover audio and a secret message, the system performs the following steps. First, the system reads the wav data of the cover audio file and checks whether there are any samples with negative values in it. If there are negative samples, those samples will be converted to positive numbers. Then, the system computes the threshold values based on the samples of the cover audio and the total number of bits in the secret message. Once the threshold values are selected, the system checks the recoverability condition. If a sample value satisfies the threshold condition and the recoverability condition, one or more bits would be embedded to the sample.

In order to retrieve the secret message, it is necessary to embed the size of the secret message. Thus, the combined secret message consists of the size and the content of the secret message. After embedding, the system converts the positive numbers, which were the negative numbers, back to negative numbers. Next, the system saves the sample values to an output file as stego audio. After successful embedding, a user can play the stego audio file to inspect its quality. Also, the system returns the threshold value T[1]

that will be needed to retrieve the secret message from the stego audio later on. We assume that the threshold valued is communicated over phone or via e-mail. It is to be noted that the goal of this project is to develop a steganographic scheme but not a cryptographic scheme. Figure 2.2 shows the embedding process.

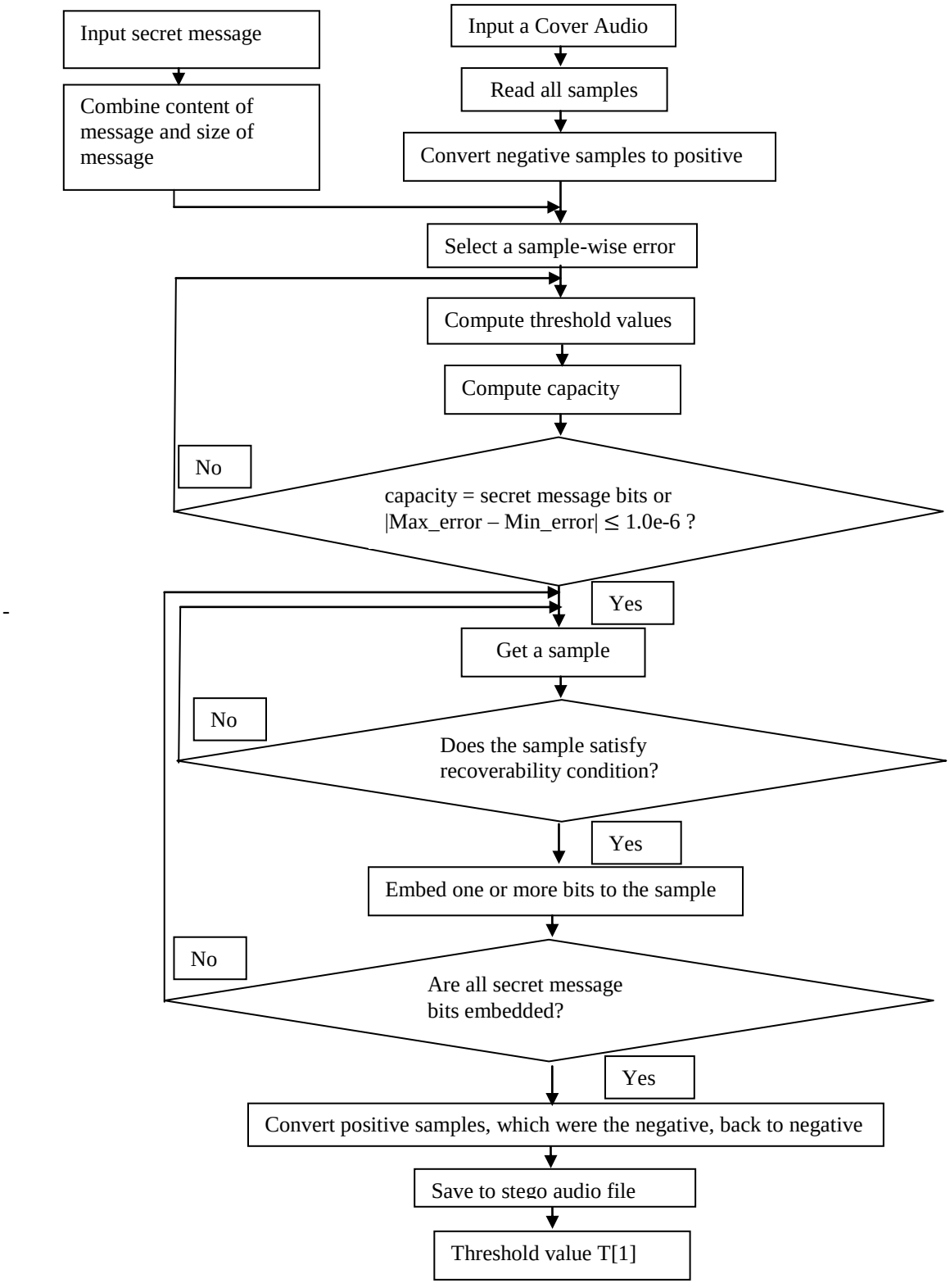

Figure 2.2 Flow Chart for Embedding Process

2.6 Retrieving Process

In order to retrieve the secret message, a user inputs a stego audio file, an output text file, and the threshold value T[1]. Then, the system reads the wav data of the stego audio file and checks whether there are any negative samples in it. All negative samples will be converted to positive numbers. Based on the threshold value, a sample-wise error δ is calculated using equation (8). Accordingly, the sample-wise error δ is:

$$\delta = \frac{1}{\lceil T[1] / 2 \rceil * 2} \tag{10}$$

After the sample-wise error δ is calculated, the other threshold values are found based on the samples of the stego audio and the sample-wise error δ. After finding the threshold values, the system checks the recoverability condition. If the samples satisfy the threshold condition and recoverability condition, the system retrieves the size of the secret message. Based on the size of the secret message, the content of the secret message is retrieved. Figure 2.3 shows the retrieving process.

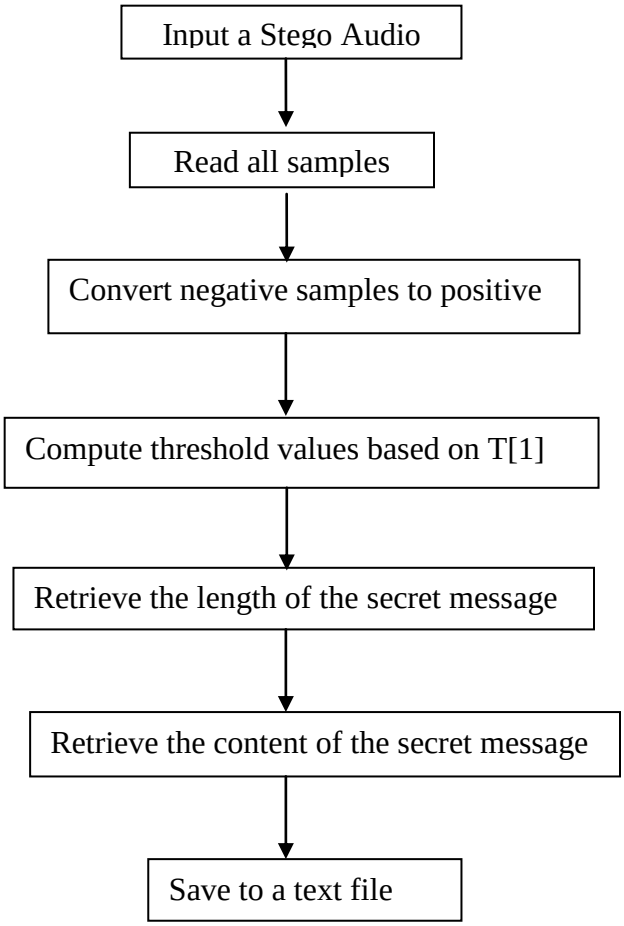

Figure 2.3 Flow Chart for Retrieving Process

3. TESTING, RESULT AND EVALUATION

This section discusses how to test the proposed scheme. In order to show the result, FFT Properties tool used to analyze the stego signals. Also, the quality sound of stego audio files is evaluated by using Signal-to-Noise Ratio (SNR), and Mean Opinion Score (MOS).

3.1 Testing

The proposed scheme is tested on four wav files which have different sampling frequencies, lengths, frequency ranges, and sizes of a cover audio. They are Belugawhale.wav, Airhorn.wav, Cockatooscreem.wav and Peacock.wav. Also the size and the content of the secret messages are diverse. The sizes of three secret messages are 3 KB, 500 bytes and 100 bytes. On the other hand, the testing shows how to use this system. The following steps show how to implement.

3.1.1 Embedding Tab

Firstly, a user clicks on table **Embed,** then click **Browse** to input a cover audio file, a secret message and output file.

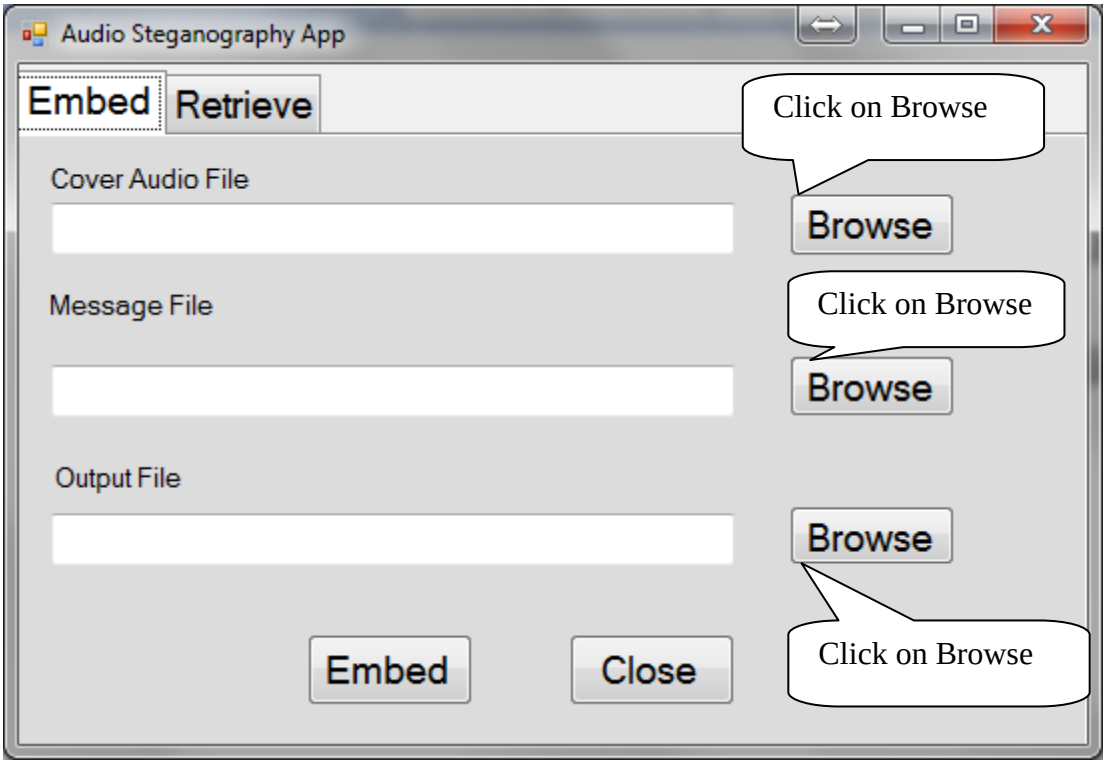

Figure 3.1 The First Step for Embedding

After inputting the cover audio file, the secret message and the output file, click on

Embed

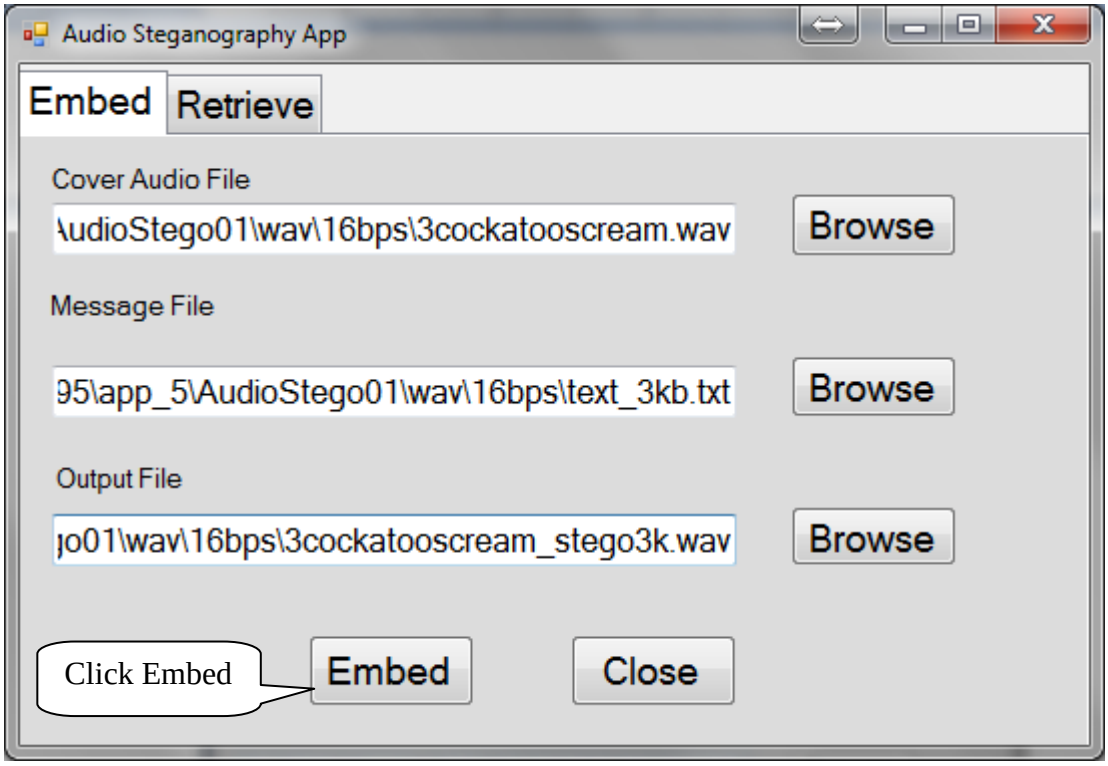

Figure 3.2 The Second Step for Embedding

In SuccessForm, click Generate Threshold, the copies the threshold value and gives a receiver. Also, the user can play or stop the audio.

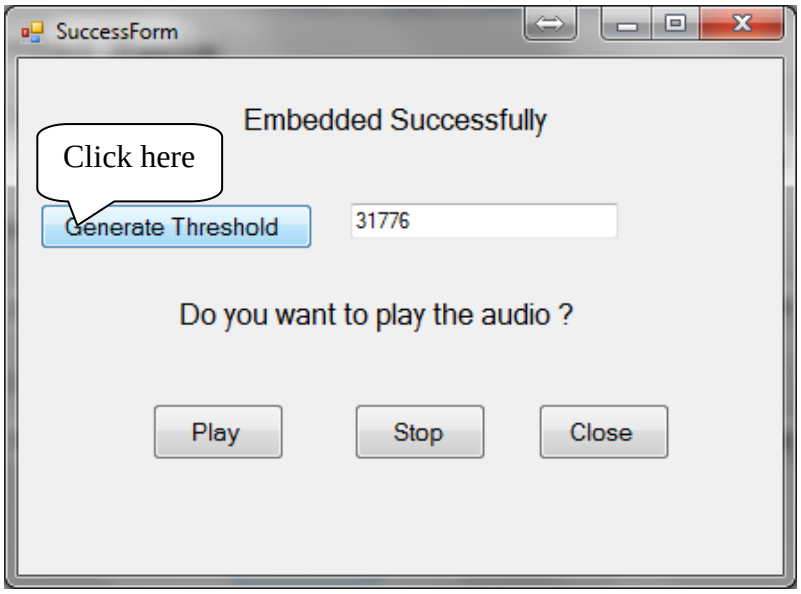

Figure 3.3 The Third Step for Embedding

3.1.2 Retrieving Tab

In order to retrieve the secret message, a user needs to choose the **Retrieve** tab, and then browse to a stego audio file path, an output file and a threshold value. Finally, click on **Retrieve**

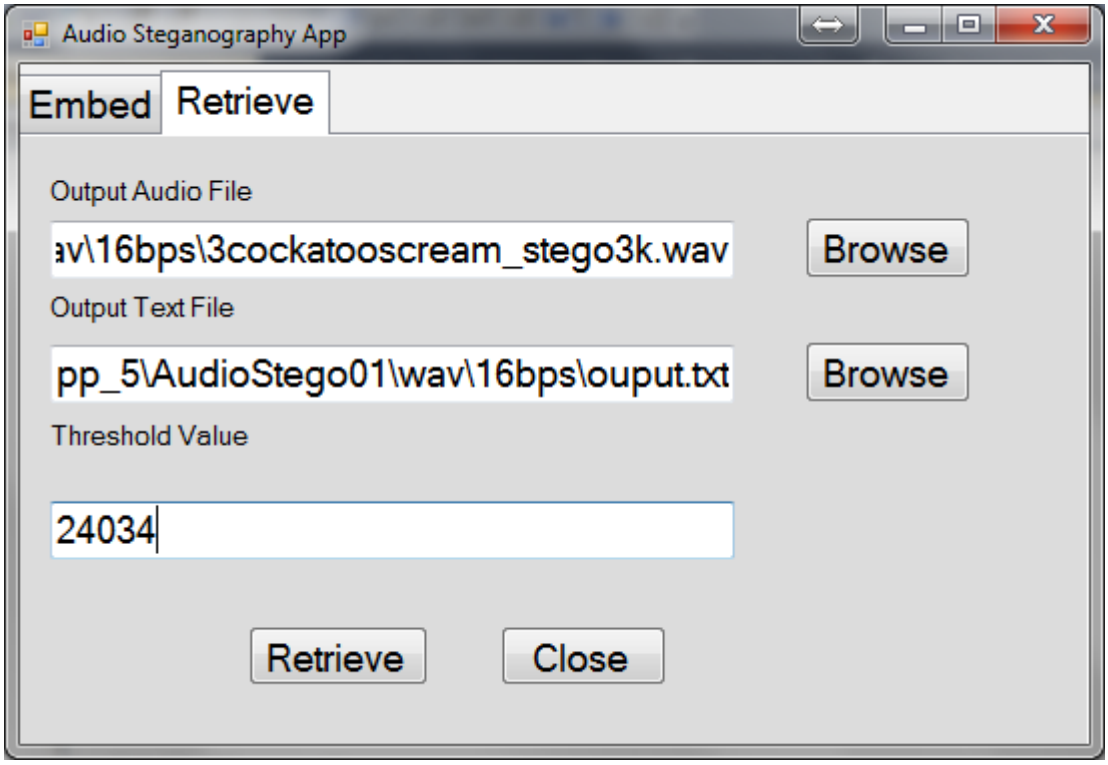

Figure 3.4 Retrieving message

The Success form will display

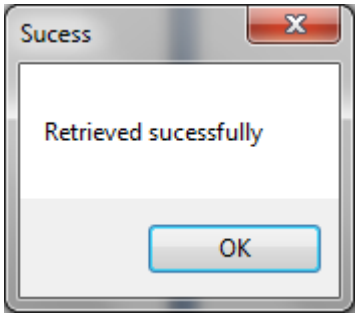

Figure 3.5 Success Form

3.2 Results

The FFT Properties tool is used to analyze the signals of the cover audio files and stego audio files. The color of the cover audio file is red while the color of the stego audio file is green. Figure 3.6, Figure 3.7, Figure 3.8, and Figure 3.9 show a small change between the cover audio file and the stego audio file. Hence, it is difficult to detect a hidden secret message in a stego audio. On the other hand, there is no difference between the secret message and the retrieved message. That means the recoverability is 100%.

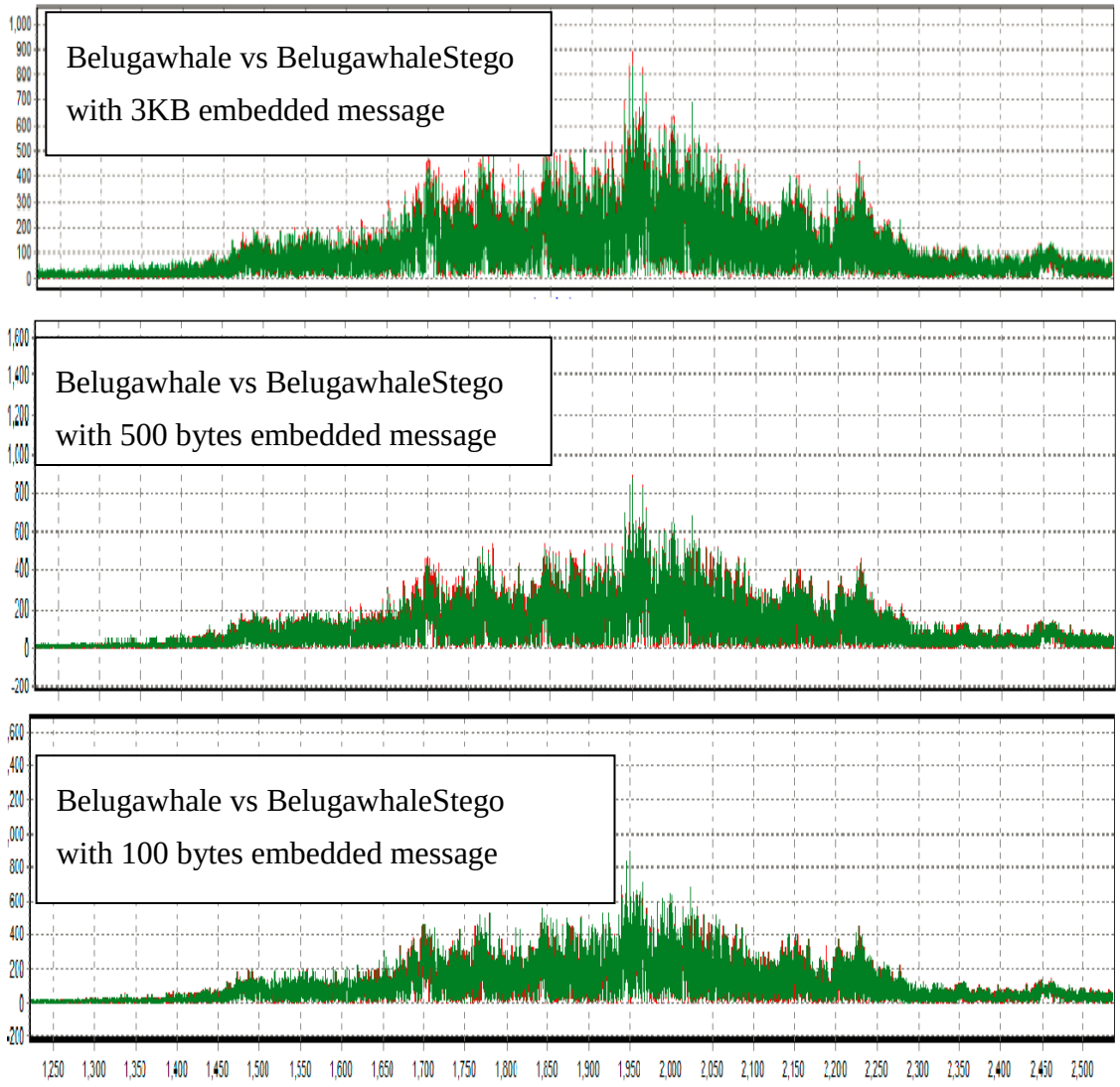

Figure 3.6 Belugawhale vs BelugawhaleStego

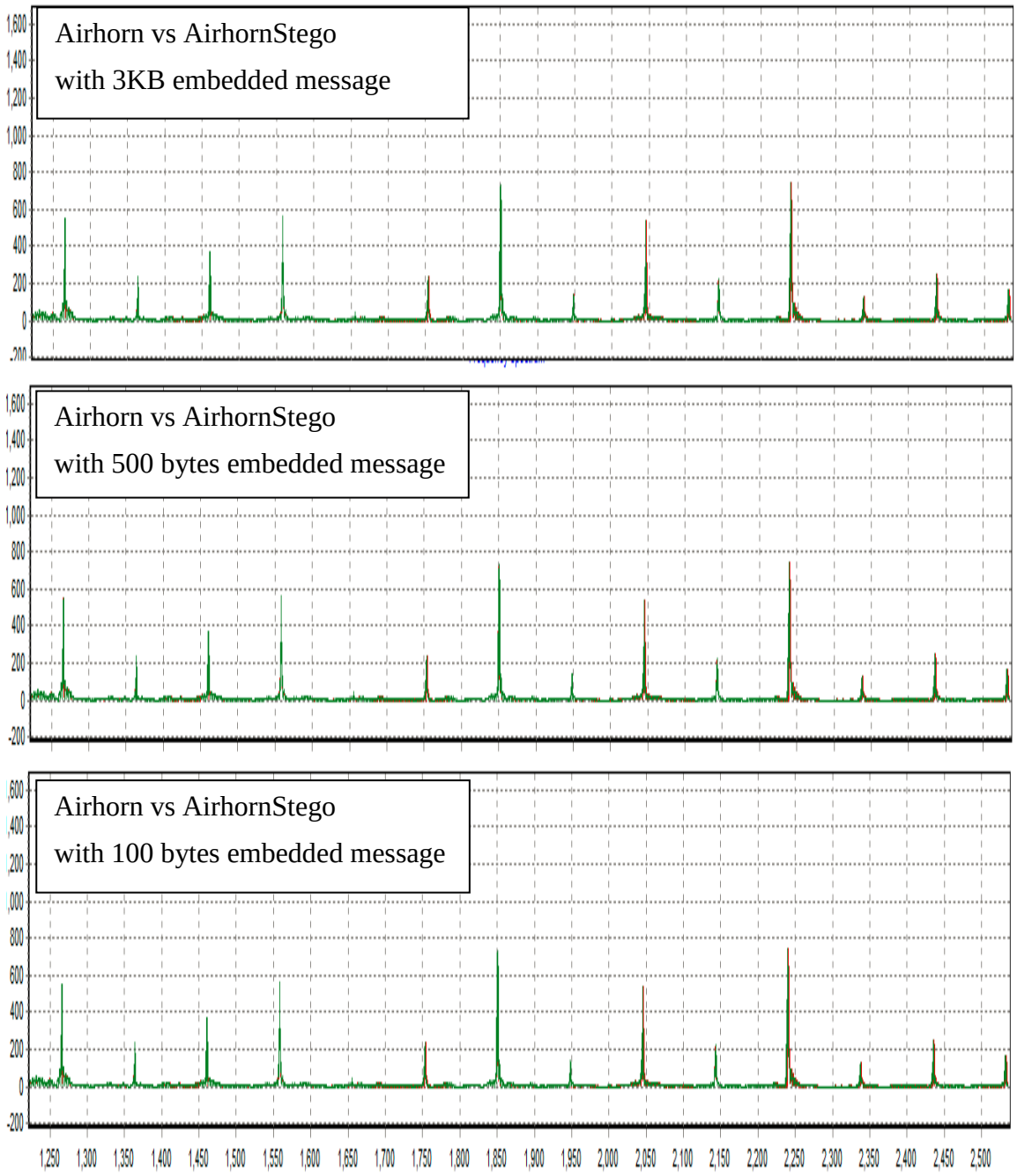

Figure 3.7 Airhorn vs AirhornStego

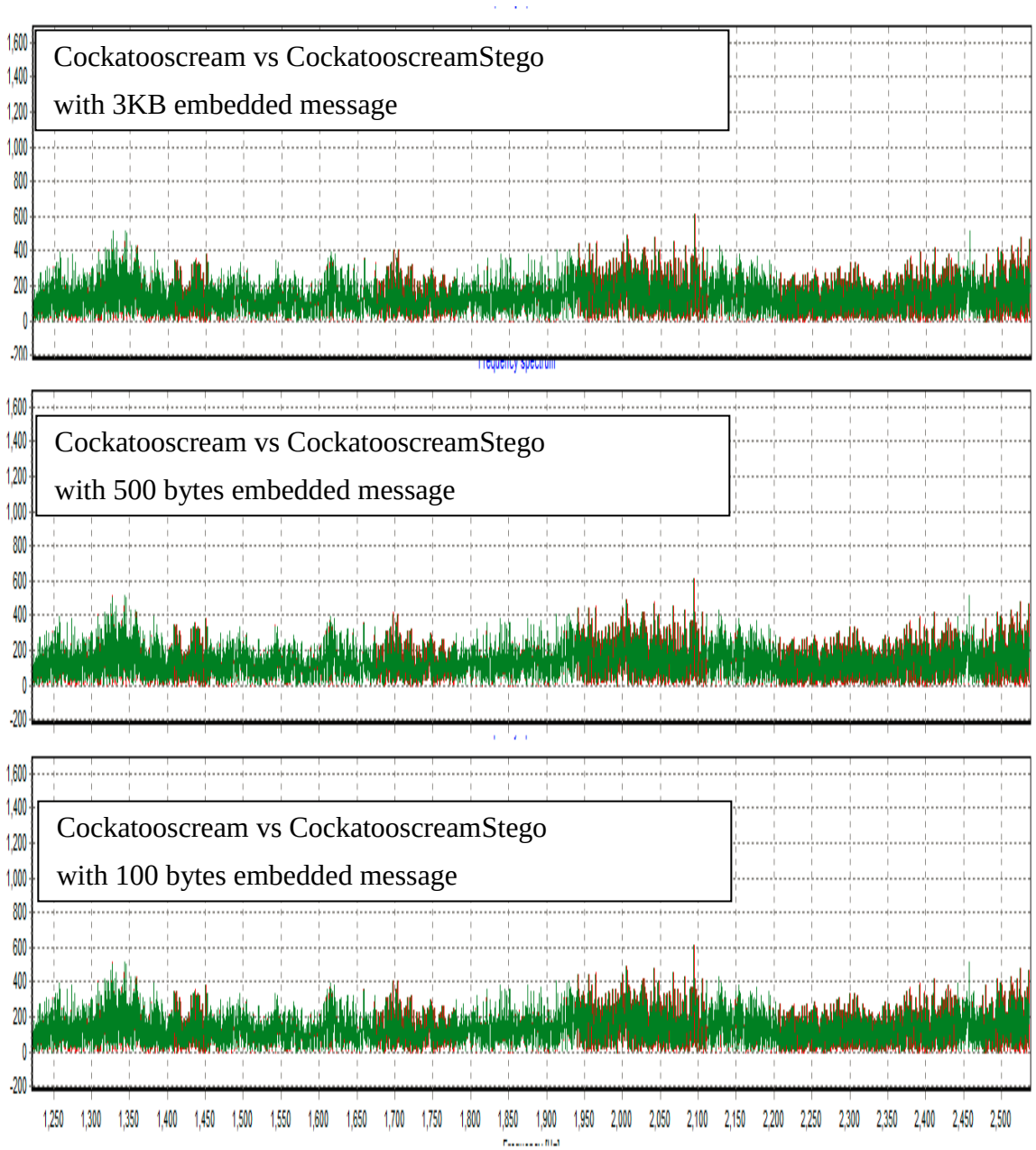

Figure 3.8 Cockatooscream vs CockatooscreamStego

Figure 3.9 Peacock vs PeacockStego

3.3 Evaluation

3.3.1 Signal Noise Ratio

The stego sounds are evaluated using Signal-to-Noise Ratio (SNR) which is defined as the ratio between cover audio signal and stego ratio signal (in dB) [8]. The bigger the SNR ratio, the better the sound quality is.

$$SNR = 10\log_{10}\left(\frac{\sum_{n}(x[n])^2}{\sum_{n}(x[n]-\hat{x}[n])^2}\right)$$

Where $x[n]$ and $\hat{x}[n]$ are the amplitude of the cover audio speech, $x(n)$ and the stego audio speed $\hat{x}(n)$ at time n respectively [3]. The Table 3.1 shows the SNR values of four different cover audio files and three diverse secret messages.

Table 3.1 SNR Values

Cover audio file	Sampling Frequency	Length	Size of cover audio	SNR (in dB) when size of secret message is:		
				100 bytes	500 bytes	3 KB
Belugawhale.wav	22 KHz	13 s	560 KB	84.18	70.47	54.89
Airhorn.wav	44 KHz	3 s	293 KB	82.62	69.32	53.37
Cockatooscream.wav	44 KHz	8 s	708 KB	88.26	74.54	58.70
Peacock.wav	44 KHz	5 s	508 KB	72.90	59.06	43.37

The Table 3.1 shows that the length, size and frequency range of the Cocktatooscreem.wav file is bigger than the Peacock.wav file; therefore, its SNR is bigger than SNR of Peacock.wav file. On the other hand, the length and size of Airhorn.wav file is smaller than Peacock.wav file, but the SNR value of Airhorn.wav is bigger than Peacock.wav because the frequency range of Airhorn.wav is higher than of Peacock.wav (Figure 3.6). It is concluded that the SNR value depends on not only the

sampling frequency, length, size, frequency range of cover audio but also the size and the content of text file.

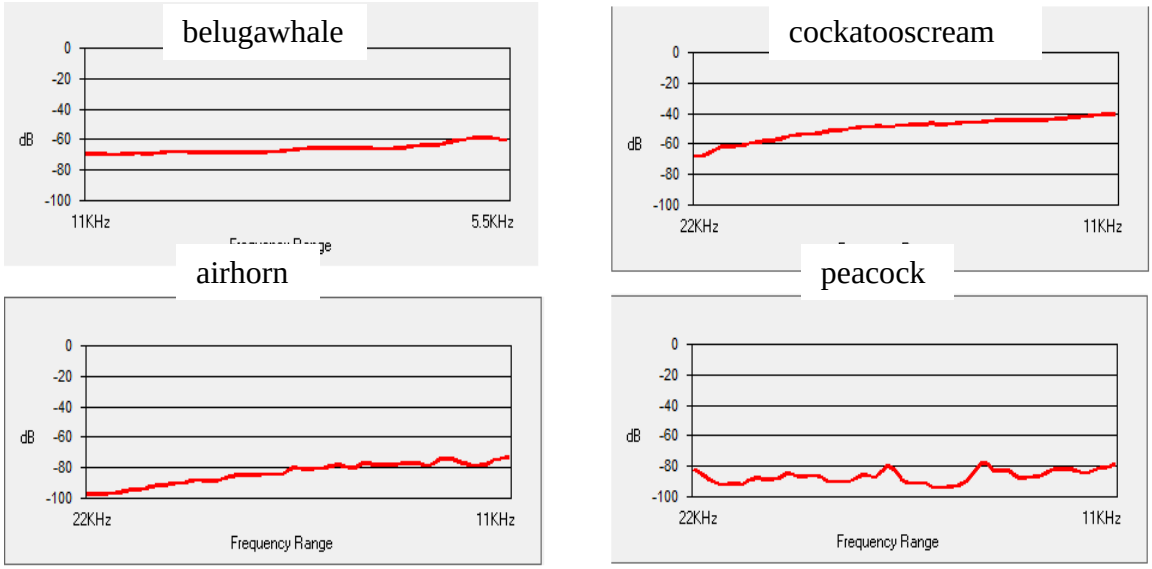

Figure 3.10 Frequency Ranges of the Cover Audio Files

3.3.2 Mean Opinion Score

Mean Opinion Score (MOS) is one of the most commonly used to evaluate the quality of sound. We asked group of five people to rate the quality of the stego sounds based on a scale 1 to 5 (1= Bad, 2 = Poor, 3 = Fair, 4 = Good, 5 = Excellent). Table 3.2 shows the MOS values of four stego audio files.

Table 3.2 MOS Values

Stego audio	The size of secret message		
	100 bytes	500 bytes	3Kbytes
Belugawhale.wav	4.2	3.6	2.8
Airhorn.wav	4.8	4.8	4.8
Cockatooscream.wav	4.8	4.4	4.2
Peacock.wav	4.6	4.6	4.6

From Table 3.1 and 3.2, it can be concluded that there are many factors affect to the quality of stego sound, such as size of cover audio, a sampling frequency, the length of a cover audio, a content and size of a secret message, etc. In order to avoid detection, a user should listen to a stego audio before sending to a receiver.

4. FUTURE WORK AND CONCLUSIONS

Information security is very important since information is exchanged on the publicly accessible Internet. Both cryptography and steganography can be used to provide information security. Steganographic techniques allow hiding valuable information in an apparently innocent document. In this work, a new audio steganographic technique has been proposed as an alternative technique to hide information in a cover audio. A well defined audio steganography scheme should satisfy at least three requirements: capacity, transparency and robustness. This scheme not only hides a secret message as big as possible but also minimizes distortion in the resultant audio as much as possible to avoid detection. This scheme can be used to embed one or many secret bits in a cover audio sample depending on the relative strength of the audio sample as the sample is measured against a scale of multi-level threshold value. The future research can combine some cryptographic techniques with this scheme. In that situation, if hackers can retrieve the secret message, they will not be able to reveal the secret message due to encryption. Another extension to this work would be to select samples chaotically instead of sequentially during embedding of the secret message bits.

BIBLIOGRPHY AND REFERENCES

[1] Asad M., Gilani J., and Khalid A., "An enhanced least significant bit modification technique for audio steganography," International Conference on Computer Network and Information Technology (ICCNIT), p. 143-147, 2011.

[2] Cvejic N., and Seppanen, "Increasing robustness of LSB audio steganography using a novel embedding method," Proceedings of International Conference on Information Technology: Coding and Computing (ITCC) P. 533- 5337, 2004.

[3] Cvejic N., and Seppanen, "Increasing robustness of LSB audio steganography by reduced distortion LSB coding," Journal of Universal Computer Science, P. 56-65, 2005.

[4] Cvejic N., and Seppanen, "A wavelet domain LSB insertion algorithm for high capacity audio steganography," Proceedings of 2002 IEEE 10th Digital Signal Processing Workshop, p. 53-55, 2002.

[5] Pooyan M., and Delforouzi A., "LSB-based audio steganography method based on lifting wavelet transform," Proceeding of IEEE Symposium on Signal Processing and Information Technology, p. 600-603, 2007.

[6] Matsuoka H., "Spread spectrum audio steganography using sub-band phase shifting," Proceedings of the International Conference on Intelligent Information Hiding and Multimedia Signal Processing, p. 3-6, 2006.

[7] Gruhl D. and Bender W., "Echo hiding," Proceeding of Information Hiding Workshop, p. 295-315, 1996.

[8] Djebbar F., Ayad B., Hammam H., and Abed-Meraim K., "A view on latest audio steganoghraphy technuiques," Proceedings of International Conference on Innovations in Information Technology (IIT) on, p. 409-414, 2011.

[9] Pareek N., Patidar V., and Sud K., "A random bit generator using chaotic maps," International Journal of Network Security, p. 32-38, 2010.

[10] Kumar H., and Anuradha, "Enhanced LSB technique for audio steganography," Proceedings of third International Conference on Computing Communication & Networking Technologies(ICCCNT), p. 1-9, 2012.

[11] Provo N., and Honeymoon P., "Detecting steganographic content on the Internet," Processing of the Network and Distributed System Security Symposium, p. 1-14, 2002.

[12] Gopalan K., "Audio steganography using bit modification," Processing of the International Conference on Multimedia and Expo, p. 629-632, 2003.