Introduction

Steganography is an art of sending hidden data or secret messages over a public channel so that a third party cannot detect the presence of the secret messages. The goal of steganography is different from classical encryption, which seeks to conceal the content of secret messages; steganography

is about hiding the very existence of the secret messages.

The advantage of steganography over cryptography alone is that the intended secret message does

not attract attention to itself as an object of scrutiny. Plainly visible encrypted messages, no matter

how unbreakable they are, arouse interest and may in themselves be incriminating in countries in

which encryption is illegal. Whereas cryptography is the practice of protecting the contents of a

message alone, steganography is concerned with concealing the fact that a secret message is being

sent as well as concealing the contents of the message

Modern steganography is generally understood to deal with electronic media rather than physical objects. There have been numerous proposals for protocols to hide data in channels containing pictures, video, audio and even typeset text. This makes sense for a number of reasons. First of all, because the size of the information is generally quite small compared to the size of the data in which it must be hidden (the cover text), electronic media is much easier to manipulate in order to hide data and extract messages.

Extraction itself can be automated when the data is electronic, since computers can efficiently manipulate the data and execute the algorithms necessary to retrieve the messages.Electronic data also often includes redundant, unnecessary and unnoticed data spaces which can be manipulated in order to hide messages. Often times we use the idea of steg-keys to remedy unwanted discrepancies. The problem with these keys is that they are needed for the final extraction of the hidden message and they have to be communicated to the receiver of the message.

The main goal of this paper is to find a way so that an audio file can be used as a host media to hide textual message without affecting the file structure and content of the audio file. Because degradation in the perceptual quality of the cover object may leads to a noticeable change in the cover object which may leads to the failure of objective of steganography. For embedding data in digital media, two domains are generally considered, spatial domain and the transform domain. Though, there are many data hiding techniques, in this paper we consider the spatial domain of data hiding.

We discuss the various data hiding techniques using bit manipulation of the lowest significant bit (LSB). We take a look at how the bit planes can be increased by various number decomposition methods without compromising on the three requirements of visibility, robustness and capacity.

1.1 Problem Definition

The information hiding system is characterised by having three different aspects that contend with each other. These are capacity, imperceptibility, and robustness. It is our goal to design robust steganography technique which can hide more data with acceptable imperceptibility of stego media. We Generate more bit planes using bit plane decomposition by different number systems, which in turn will provide greater capacity that could be use to conceal more data without causing significant distortion, providing greater security against steganalysis programs.

1.2 Objective

It is our goal to design robust steganography technique which can hide more data with acceptable imperceptibility of stego media. The main objective is:

1. Embedding data in uncompressed audio files based on spatial domain.

2. Enhance the capacity by generating more number of bit planes using bit plane decomposition technique using various number systems.

3. Use double layer security using any extra files to carry key.

4. Enhance robustness by embedding encrypted data at multiple bit planes.

1.3 Tools and Platform

· Python 3.6.7 (Apple Inc. 4.2.1 build 5666)

· MacOS El Capitan

· Hard Disk : 256 TB

· RAM : 8 GB

· Wavepad Audio editor

1.4 Project Timeline

?

Literature Survey

The simplest implementation of data hiding technique is the classical least significant bit data hiding technique [2]. The technique is based on manipulation of the least significant bits of the carrier image to accommodate the hidden message. The insertion of LSB varies according to the number of bits in an image. For an 8 bit image, the value of 8th bit, which is the least significant bit of each pixel, would be modified and the secret message would be embedded [3].Suppose if we wanted to insert the letter ‘A’ into an image. The binary equivalent of ‘A’ is 10000001. Now if our sample image of 8 bit has the following pixel values:

10101010 10001010 11111111 010110101

10101010 10101010 10001011 10101111

Battisti et al.[1] proposed a method of embedding data into digital media by decomposition of Fibonacci number sequence which allowed different bit plane decomposition when compared to the classical LSB scheme. The Fibonacci sequence, named after Leonardo of

Pisa, also known as Fibonacci, is a sequence of numbers in the following integer sequence:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89 ... The bit planes are decomposed based on the Fibonacci sequence. The main drawback of this approach is that of redundancy and to counter that and obtain a unique representation, Zeckendorf theorem is used. To embed the intended message in the cover image, it is decomposed into bit planes by using Fibonacci decomposition. The Zeckendorf condition is checked for each bit to be modified. If the condition is fulfilled, the bit is inserted otherwise the bit following it is considered.

LSB data hiding using prime numbers is a data hiding technique proposed by Dey et.al [4] as improvement over the Fibonacci numbers data hiding technique proposed by Battisti et. al.[1] The main idea the work was to use the prime number decomposition generate new set of bit planes and em information in these newly generated bit planes minimal distortion. In this approach, the researchers took an image obits and increased the number of bit planes there the value of n was equal or greater the equal to the number of bit planes of the image. was achieved by converting the bit planes of image to another number system using numbers as the weighted function. This resulted in increase of number of bits and consequently it cbe used for hiding data in higher bit planes minimal distortion. For decomposition, the weight function was defined as: P (0) = 1, P (i) = pi ∀ i ∈ Z+ , pi = ith Prime In case of any ambiguity, the lexicographically higher number is given preference.

The approach of LSB data hiding by natural numbers was proposed by Dey et.al.[5] In this approach, the researchers proposed data hiding by decomposition of a pixel value in sum of natural numbers. This resulted in generation of more bit planes than the Classical LSB data hiding, Fibonacci LSB data hiding and the Prime number data hiding . For decomposition, the weight function is defined as: W (i) = N (i) = i + 1, ∀ I ∈ Z+ ∪ {0} .The researchers used the same concept in case of ambiguity which gave higher precedence to lexicographically higher number. For embedding the data into the k‐bit image, a number n is chosen in a way such that all pixel values in the range of [0, 2k – 1] could be represented using first n numbers, which resulted in generation of n virtual bit planes .After finding the value of n, a k bit to n bit map is created and all valid representations in natural numbers system are marked. In case of any ambiguity, the lexicographically higher number is given preference.

[6] The paper discuss generation an entirely new set of bit planes and embed data bit in these bit planes, using decomposition techniques. For convenience of description, here, the LSB is called the 0 th bit, the second LSB is called the 1 st bit, and so on. The newly-generated set of bit-planes ’virtual’, since these bit-planes are not obtain in classical binary decomposition of technique. The embedding technique in classical LSB and Fibonacci decomposition technique with new modification. A generalized approach that follow novel data-hiding techniques using prime/natural number decomposition. The embedding technique using the prime decomposition. Experimental results obtained using this technique are reported. It is shown (both theoretically and experimentally) that the data-hiding technique using prime decomposition out performs the famous LSB data hiding technique using classical binary decomposition and that using Fibonacci p-sequence decomposition. Also, the technique using natural number decomposition out performs the one using prime decomposition, when thought with respect to embedding secret data bits at higher bit-planes (since number of virtual bit-planes generated also increases) with less detectable distortion.

[7] This paper proposed a substitutive steganographic technique that is performed in a novel domain based on the generalized (p, r) − Fibonacci sequence F p,r (n). Experimental results show the effectiveness of the proposed method and the improvement with respect to the classical LSB decomposition both for security and perceptual aspects.

[8]In this paper of embedding text-based data into a host audio file using the method of bit modification has been presented . A procedure has been developed in which the data field is edited to embed intended data into the audio file. To proceed with this, the header section of the audio has been checked perfectly because a minimal change in the header section may leads to a corruption of whole audio file. The main goal of this 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. In test cases the text-based data has been successfully embedded to the audio file to visualize in what extent the target has been achieved. However future scope

appears end less.

[9]This paper discuss the idea of a number system using combination of catalan number and general fibonacci series that can generate more comprehensive bit planes.This paper shows using Catalan numbers is an improvement over the Fibonacci data hiding technique. In the classical LSB technique, it is possible to embed secret data only in the

first few bit planes, since image quality becomes drastically distorted when embedding data in higher bit planes. This method uses a bigger set of virtual bit planes, increased number of stego bits that can be embedded. Besides, apart from the LSB method, in which the tracking of data is easy (if there is suspicion of course), the Fibonacci and Catalan methods provide a kind of encryption, via the way that they hide the bits in more bit planes. Specifically, this method employs a larger number of bits lanes, offering not only more space for our stego data, but also greater security against steganalysis programs.

Concept and problem analysis

Design and Methodology.

The implementation of the used algorithms are done in a modular fashion. The codes for the proposed methods are quite similar barring few key differences. Modular methods of writing codes helps in the reusability of the codes making it simple and easy to use for further purposes.We have created four modules namely, Extraction module, embedding module, bit planes module, compressor module

| table 1 : Table representing all the used number system | | | | | | | | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| prime | 1 | 2 | 3 | 5 | 7 | 11 | 13 | 17 | 19 | 23 | 29 | 31 | 37 | 41 | 43 |
| natural | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| fibo | 1 | 2 | 3 | 5 | 8 | 13 | 21 | 34 | 55 | 89 | 144 | 233 | 377 | 610 | 987 |
| fib(2) | 1 | 2 | 3 | 4 | 5 | 6 | 9 | 13 | 19 | 28 | 41 | 60 | 88 | 129 | 189 |
| fib(1,2) | 1 | 1 | 1 | 2 | 2 | 4 | 4 | 8 | 8 | 16 | 16 | 32 | 32 | 64 | 64 |
| catalan | 1 | 2 | 3 | 5 | 8 | 13 | 14 | 21 | 34 | 42 | 55 | 89 | 132 | 144 | 233 |

.

4.1 Implementation of bit planes module

Bit planes modules works as a storage module that provide with all the number systems that is used for proposed methods. It uses the following table.

4.1.1 Bit planes sub module for representing numbers in binary form of different number systems.

Step 1 : Start

Step 2 : The proposed number system is taken according to the need from the given table.

Step 3: An binary array is made and initialize with zeros. The size of the array is defined according to the bit depth of the sample cover file and the number system used.

Step 4: Input is taken as n.

Step 5: Closed number to n is taken from the number system.

Step 6: The face value is made one in the binary array corresponding to the closed number from step 5.

Step 7 : Difference between the closed number and n is taken and saved in f.

Step 8: Change to value of n = f

Step 9 : Back to step 5 till n >= 0.

4.1.2 Bit planes sub module for representing binary form numbers back to decimal form.

Step 1 : Start

Step 2 : The proposed number system is taken according to the need from the given table.

Step 3: A binary array is taken as input arr.

Step 4: All the elements of the array is probed and the place values containing one are noted.

Step 5: The values of the place values are add. The final result if the required result

4.2 Implementation of Embedding module.

The embedding module is divided into two sub module. One module implements data hiding using 2 bit planes and another module implements data hiding using 3 bit planes.

4.2.1 Embedding sub module for data hiding using 2 bit planes.

Step 1 : The cover audio file is taken as input.

Step 2 : The secret message is taken as input.

Step 3 : The Satadhi’s cipher is applied into secret message for extra layer of abstraction.

Step 4 : The secret message is stored as steams of bits using utf-8 format.

Step 5 : Each samples of the audio file is represented in binary form according to the proposed number system used.

Step 6 : Two bit planes are chosen which are closed to least significant bit planes and does not have bottleneck.

Step 7 : the number of samples in the cover media > The of bits in the steams of bits of the secret message +4 bytes (for extra infromation) if not then exception is put to input bigger cover audio file.

Step 8 : The Stego-key is generated according to the samples that are used for saving the hiddien message.

Step 9 :The audio file that was in binary form is converted back to decimal form.

Step 10: The cover audio file is reconstructed.

Step 11 : Compressor module is used to compress the stego-key.

4.2.2 Embedding sub module for data hiding using 2 bit planes.

Step 1 : The cover audio file is taken as input.

Step 2 : The secret message is taken as input.

Step 3 : The Satadhi’s cipher is applied into secret message for extra layer of abstraction

Step 4 : The secret message is stored as steams of bits using utf-8 format.

Step 5 : Each samples of the audio file is represented in binary form according to the proposed number system used.

Step 6 : Three bit planes are chosen which are closed to least significant bit planes and does not have bottleneck.

Step 7 : the number of samples in the cover media > The of bits in the steams of bits of the secret message +4 bytes (for extra information) if not then exception is put to input bigger cover audio file.

Step 8 : The Stego-key is generated according to the samples that are used for saving the hidden message.

Step 9 :The audio file that was in binary form is converted back to decimal form.

Step 10: The cover audio file is reconstructed.

Step 11 : Compressor module is used to compress the stego-key.

4.3 Implementation of Extracting module.

The extracting module is divided into two sub module. One module implements extracting algorithm from cover media that used 2 bit planes for embedding and another module implements extracting algorithm for cover media that used 3 bit planes.

4.3.1 Extracting sub module for data extracting for 2 bit planes.

Step 1 : The reconstructed cover audio file is taken as input.

Step 2 : The stego-key is also taken as input.

Step 3 : Compressor module is used to decompress the stego-key.

Step 4 : The last 4 bytes are extracted first as it contains the information required for extraction of secret message.

Step 5 : The samples from the cover audio file is chosen according to the stego-key for the extraction of the message.

Step 6 : The steams of bits are taken from the 2 used bit planes for hiding data.

Step 7 : The steams of bits are then reconstructed using utf-8 format .

4.3.2 Extracting sub module for data extracting for 3 bit planes.

Step 1 : The reconstructed cover audio file is taken as input.

Step 2 : The stego-key is also taken as input.

Step 3 : Compressor module is used to decompress the stego-key.

Step 4 : The last 4 bytes are extracted first as it contains the infromation requied for extraction of secret message.

Step 5 : The samples from the cover audio file is chosen according to the stego-key for the extraction of the message.

Step 6 : The steams of bits are taken from the 3 used bit planes for hiding data.

Step 7 : The steams of bits are then reconstructed using utf-8 format .

4.4 Implementation of Compressor Module

The compressor module has two sub module namely compress and decompress.

4.4.1 compress sub module

Step 1 : The stego-key is taken as input.

Step 2 : A base value is initialized which is equal to the first value of the stego-key.

Step 3 : A array is declared as temp\_stego\_key.

Step 4 : From the start two consecutive stego value is taken and subtracted from in that order.

Step 5 :The value after subtraction is stored in the temp\_stego\_key chronologically.

Step 6 : Repeat the process till the end of stego-key.

Step 7 : Store the base value at the end of temp\_stego\_key.

4.4.1 decompress sub module

Step 1 : the compressed stego-key is taken as input.

Step 2 : The base value is extracted from the compressed stego-key.

Step 3 : A array is declared as actual\_stego\_key.

Step 4 : Add the base value to the first value of the compressed stego-key file and save it in the actual\_stego\_key.

Step 5 : Add the corresponding values of the compressed key with the previous actual\_stego\_key value. Continue till the compressed stego-key ends.

Step 6 : The actual\_stego\_key is the actual stego-key after decompression.

5 Testing, Results, Discussion on Results

The performance of the algorithm is evaluated based on three parameters: imperceptibility, security and capacity. In any Steganography technique these three performance evaluators play vital role. The imperceptibility is ability of Steganography technique to bring the difference between cover image and stego image which should be near to zero. Here, imperceptibility is achieved by having unnoticeable difference

between cover audio and stego audio. The capacity is increase in significant proportion.

Terminology and Definition.

SNR :- A signal-to-noise ratio compares a level of signal power to a level of noise power. It is most often expressed as a measurement of decibels (dB). Higher numbers generally mean a better specification, since there is more useful information (the signal) than there is unwanted data (the noise).For example, when an audio component lists a signal-to-noise ratio of 100 dB, it means that the level of the audio signal is 100 dB higher than the level of the noise. A signal-to-noise ratio specification of 100 dB is considerably better than one that is 70 dB (or less).

| Table 2: Results for audio1.wav | | | | |
| --- | --- | --- | --- | --- |
| Number systems | SNR | sample’s probed | samples’s used | SPCC |
| prime | 25.9089 | 18543 | 8474 | 0.99993250 |
| catalan | 27.415 | 14161 | 8474 | 0.9997186 |
| fibonacci | 27.6889 | 13462 | 8474 | 0.9996611 |
| fib-(2) | 22.2649 | 16572 | 8474 | 0.99989373 |
| fib(1,2) | 31.240 | 8474 | 8474 | 0.9968333 |

Squared Pearson Correlation Coefficient (SPCC): Squared Pearson Correlation Coefficient (SPCC) is also used to measures the similarity level between these two signals. The higher the SPCC level, the better is the similarity level. The value of SPCC should be in range of 0 to 1.

Results

5.1 Results for 2 bit planes data hiding in different audio files.

5.1.1 Result and analysis on Audio1.wav

A sample Audio.wav is taken and all the above mention data hiding technique is used .

Fig 1(a) is the original audio1.wav cover file used for experiment.

Fig 1(b) is the reconstructed audio1.wav after embedding data in 2 bit planes using bit plane decomposition by catalan number system.

Fig 1(c) is the reconstructed audio1.wav after embedding data in 2 bit planes using bit plane decomposition by prime number system.

Fig 1(d) is the reconstructed audio1.wav after embedding data in 2 bit planes using bit plane decomposition by fibonacci number system.

Fig 1(e) is the reconstructed audio1.wav after embedding data in 2 bit planes using bit plane decomposition by fibonacci(2) system.

Fig 1(f) is the reconstructed audio1.wav after embedding data in 2 bit planes using bit plane decomposition by fibonacci(1,2) system.

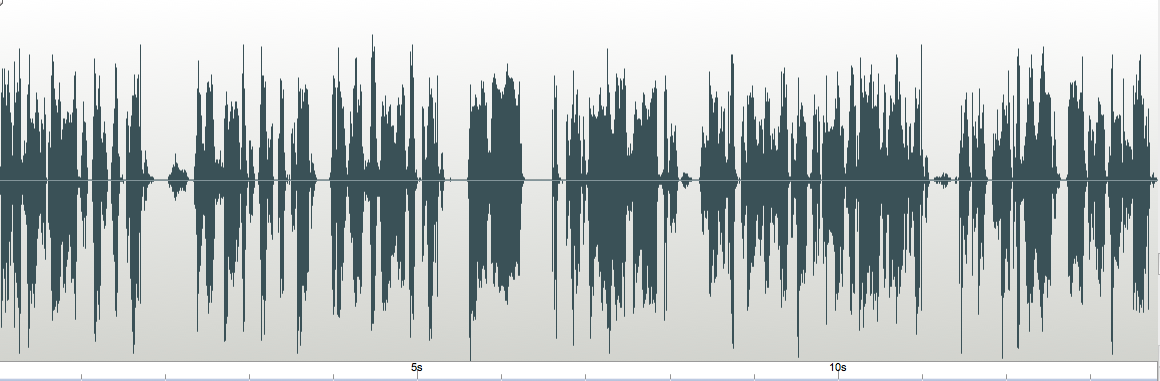
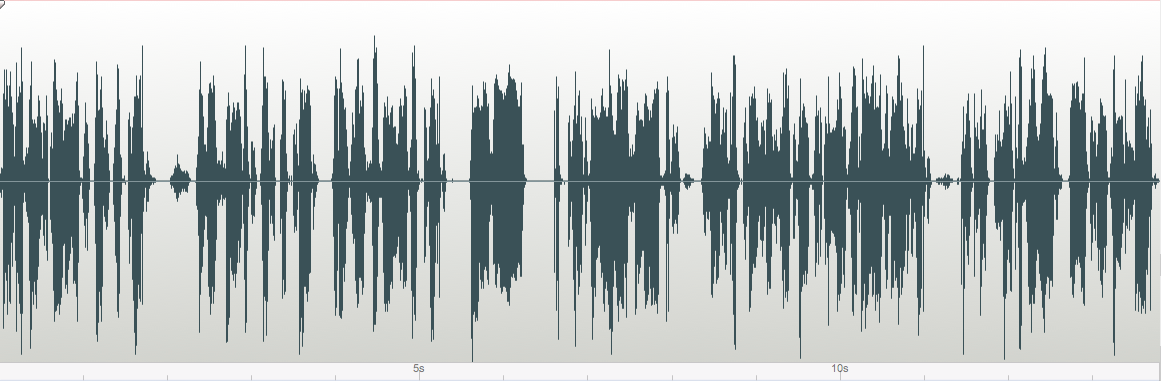


Fig 1(a)

Fig 1(b)

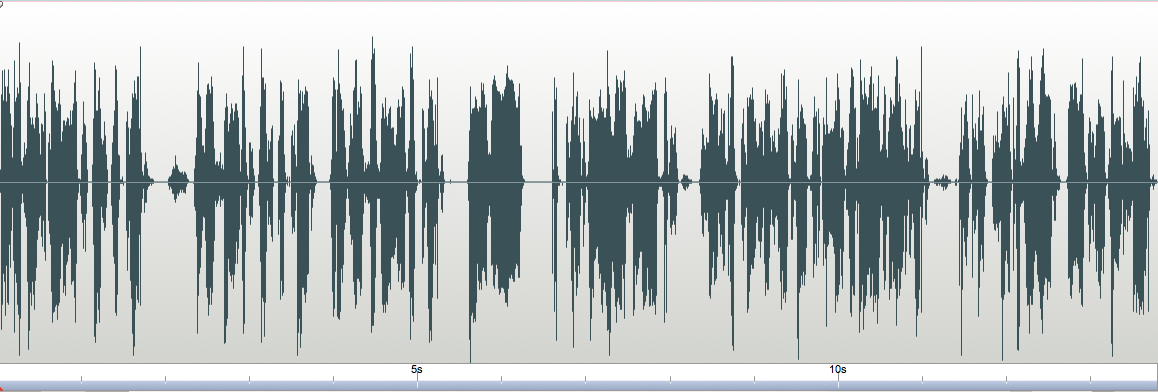
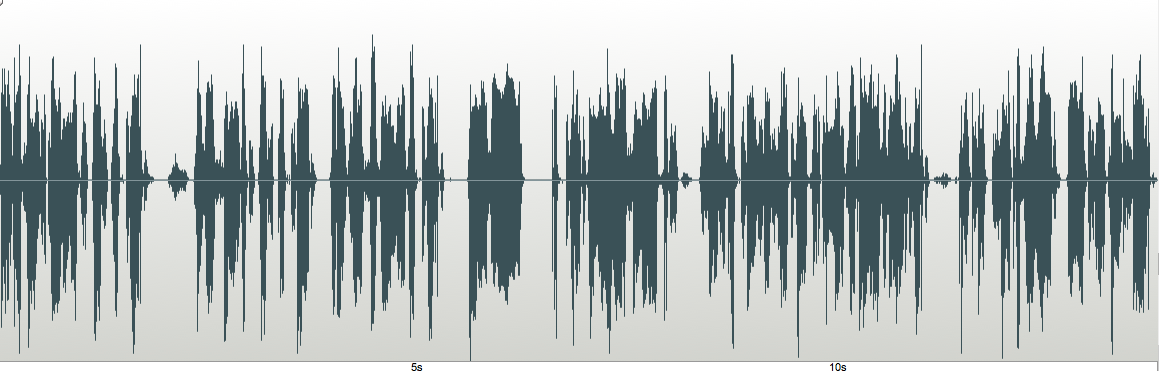


Fig 1(c)

Fig 1(d)

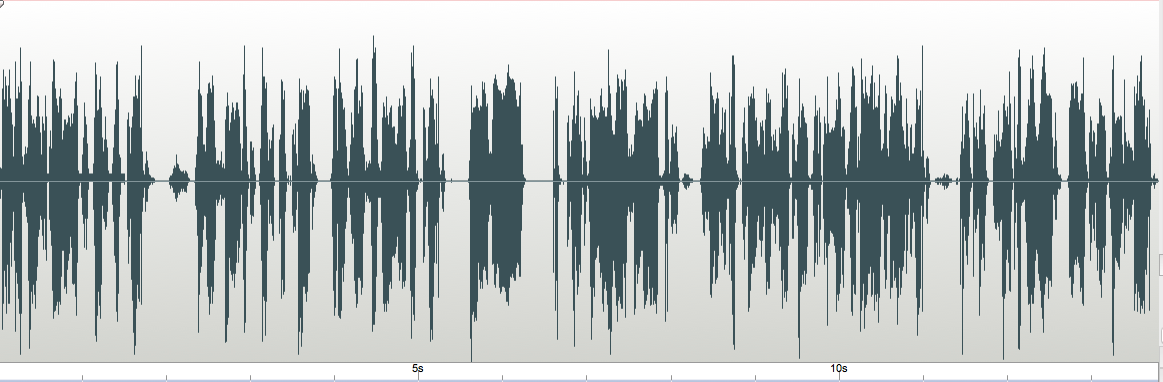
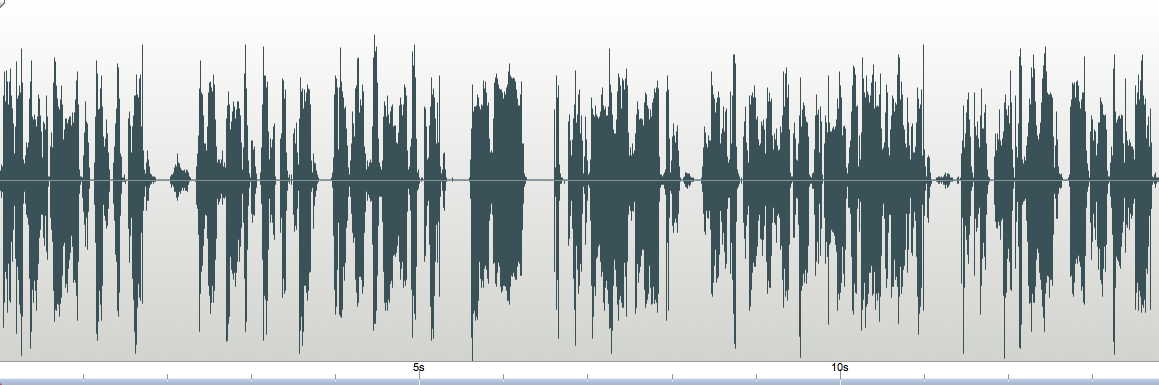


Fig 1(e)

Fig 1(f)

5.1.2 Result and analysis on Audio2.wav

Fig 2(a) is the original audio1.wav cover file used for experiment.

Fig 2(b) is the reconstructed audio2.wav after embedding data in 2 bit planes using bit plane decomposition by catalan number system.

Fig 2(c) is the reconstructed audio2.wav after embedding data in 2 bit planes using bit plane decomposition by prime number system.

Fig 2(d) is the reconstructed audio2.wav after embedding data in 2 bit planes using bit plane decomposition by fibonacci number system.

| Table 3: Results for audio2.wav | | | | |
| --- | --- | --- | --- | --- |
| Number systems | SNR | samples probed | samples used | SPCC |
| prime | 24.4061 | 14357 | 8474 | 0.99979 |
| catalan | 29.902 | 11795 | 8474 | 0.99966 |
| fibonacci | 32.1322 | 12083 | 8474 | 0.99968 |
| fib-(2) | 23.758 | 17679 | 8474 | 0.99989 |
| fib(1,2) | 24.9799 | 8474 | 8474 | 0.99837 |

Fig 2(e) is the reconstructed audio2.wav after embedding data in 2 bit planes using bit plane decomposition by fibonacci(2) system.

Fig 2(f) is the reconstructed audio2.wav after embedding data in 2 bit planes using bit plane decomposition by fibonacci(1,2) system.

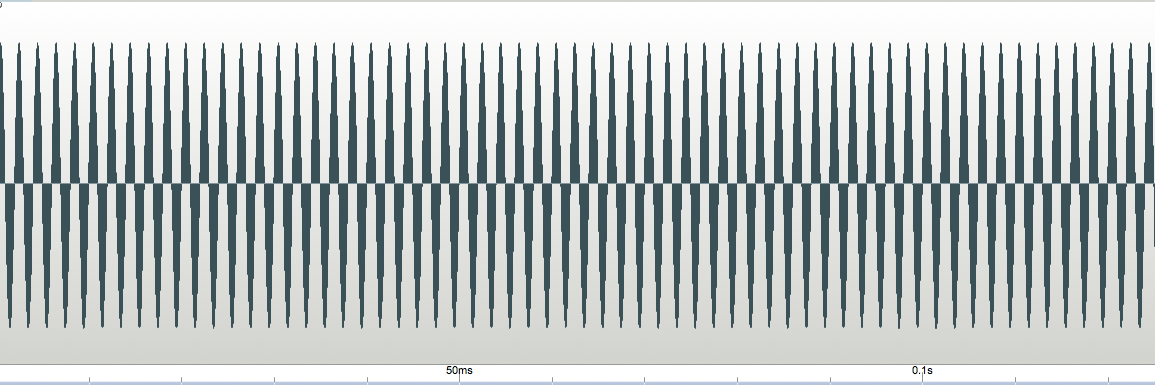
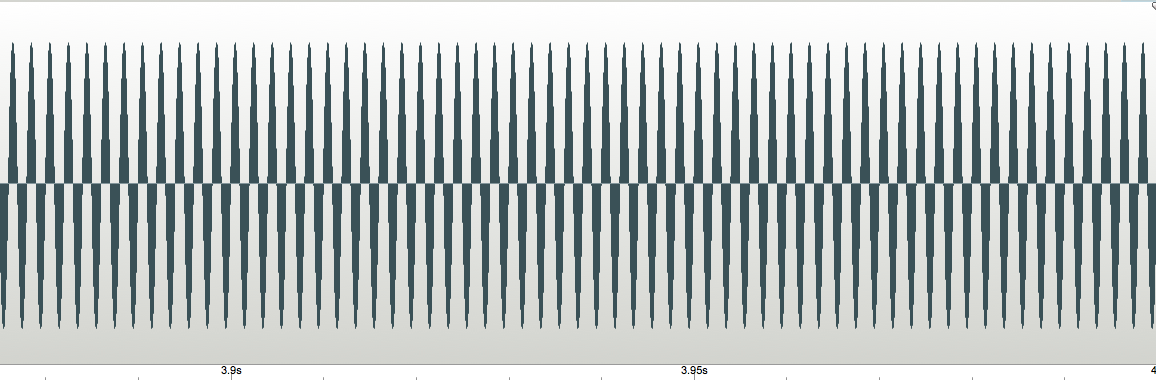


Fig 2(a)

Fig 2(b)

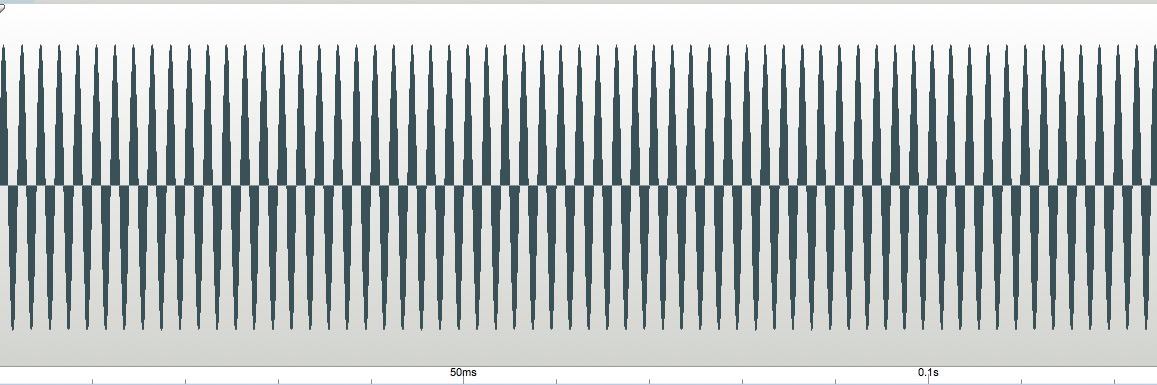
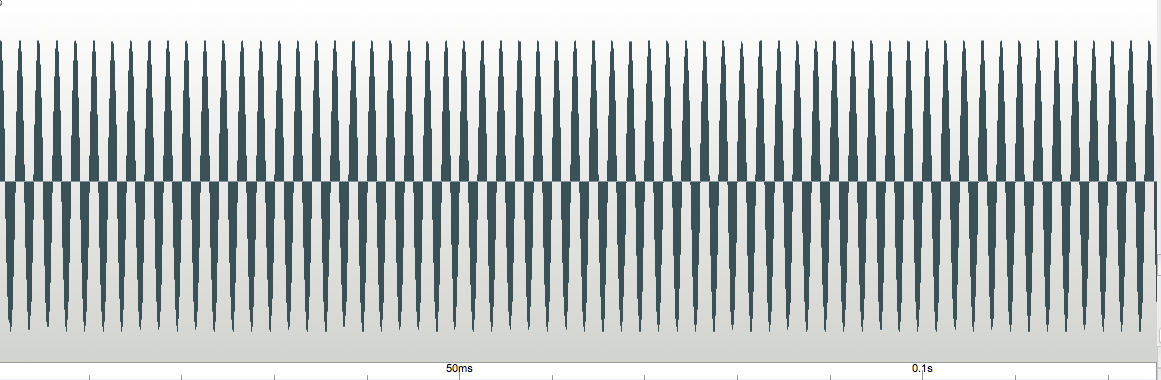


Fig 2(c)

Fig 2(d)

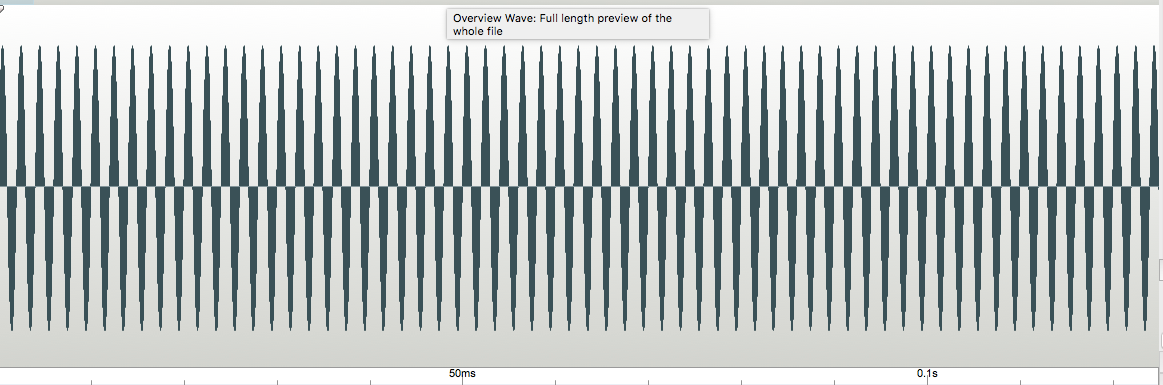
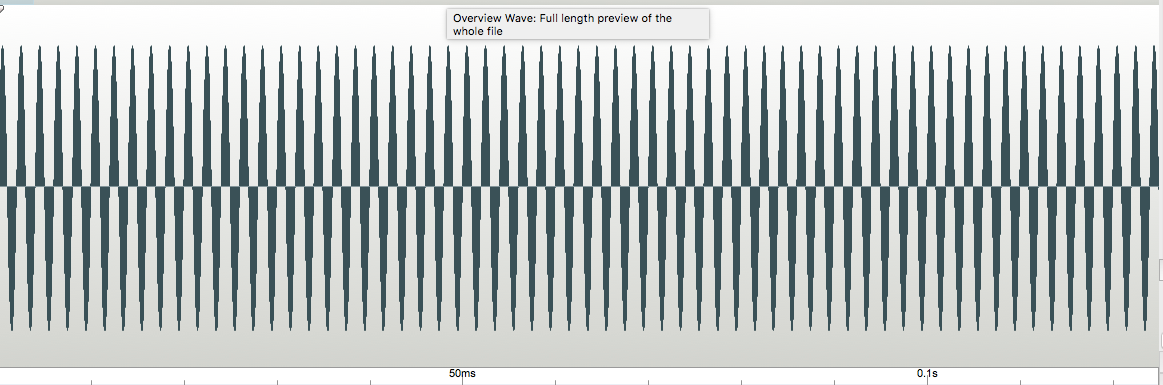


Fig 2(e)

Fig 2(f)

5.1.2 Result and analysis on Audio3.wav

A sample Audio.wav is taken and all the above mention data hiding technique is used .

Fig 1(a) is the original audio1.wav cover file used for experiment.

Fig 1(b) is the reconstructed audio1.wav after embedding data in 2 bit planes using bit plane decomposition by catalan number system.

Fig 1(c) is the reconstructed audio1.wav after embedding data in 2 bit planes using bit plane decomposition by prime number system.

| Table 4: Results for audio3.wav | | | | |
| --- | --- | --- | --- | --- |
|  | SNR | total sample | used | spcc |
| prime | 28.0401 | 14888 | 8474 | 0.99988 |
| catalan | 26.854 | 13390 | 8474 | 0.99976 |
| fibonacci | 27.320 | 13248 | 8474 | 0.9975 |
| fib-(2) | 24.5788 | 17410 | 8474 | 0.99987 |
| fib(1,2) | 26.2080 | 8474 | 8474 | 0.99954 |

Fig 1(d) is the reconstructed audio1.wav after embedding data in 2 bit planes using bit plane decomposition by fibonacci number system.

Fig 1(e) is the reconstructed audio1.wav after embedding data in 2 bit planes using bit plane decomposition by fibonacci(2) system.

Fig 1(f) is the reconstructed audio1.wav after embedding data in 2 bit planes using bit plane decomposition by fibonacci(1,2) system.

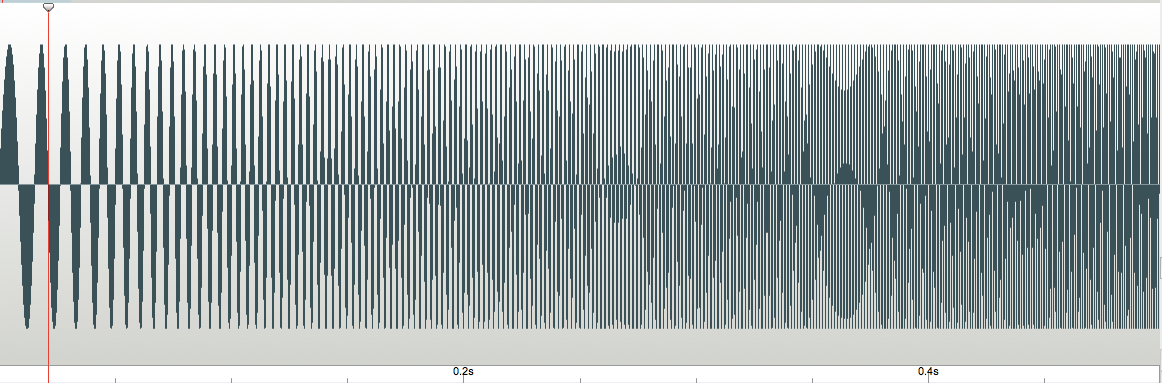


Fig 2(c)

Fig 2(c)

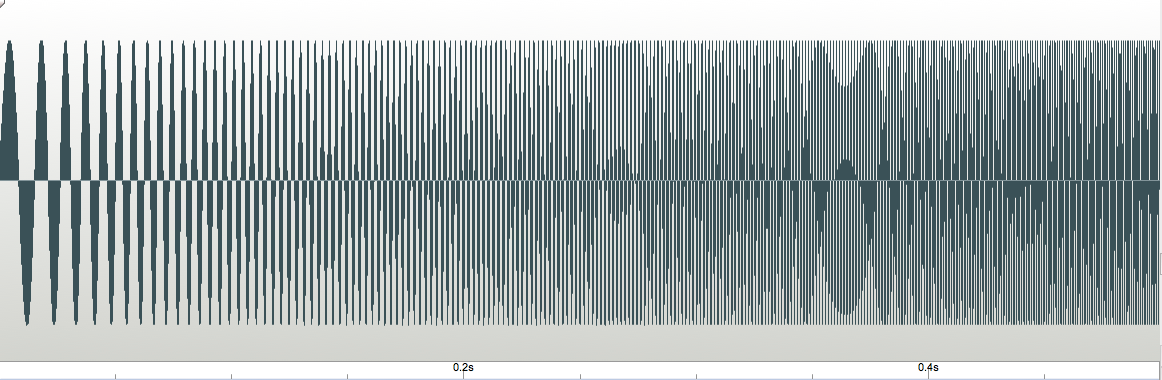
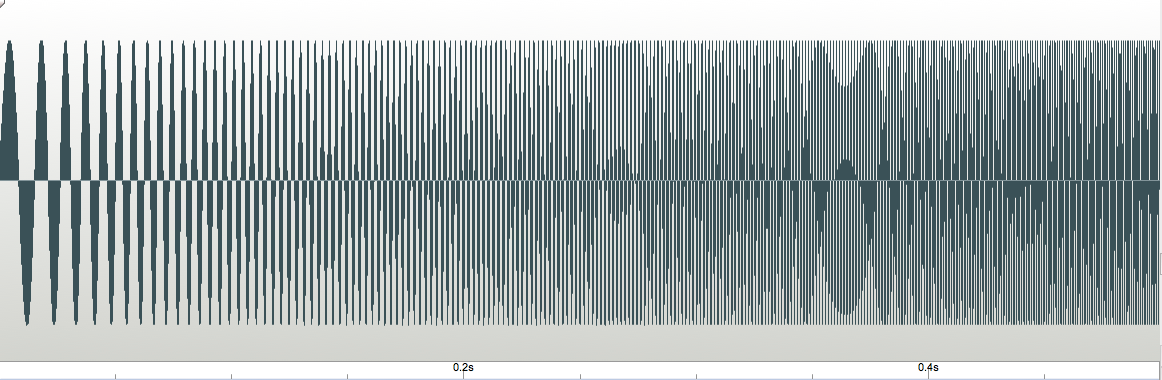
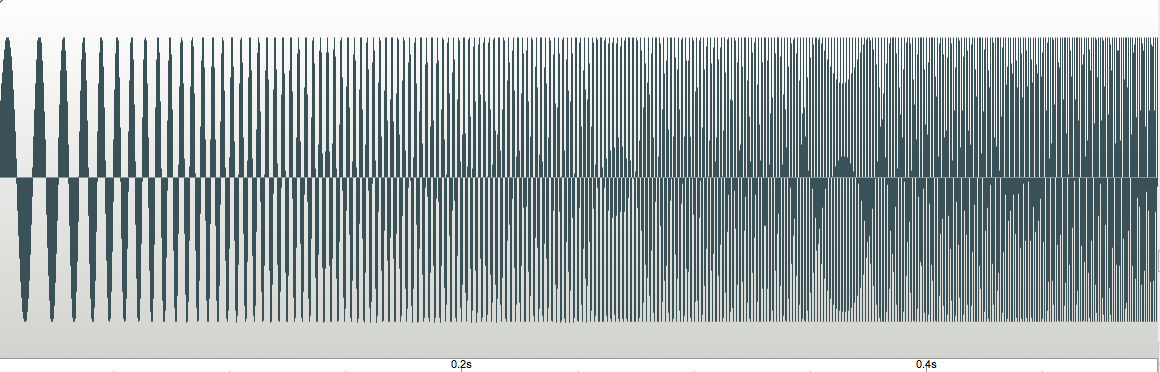


Fig 2(c)



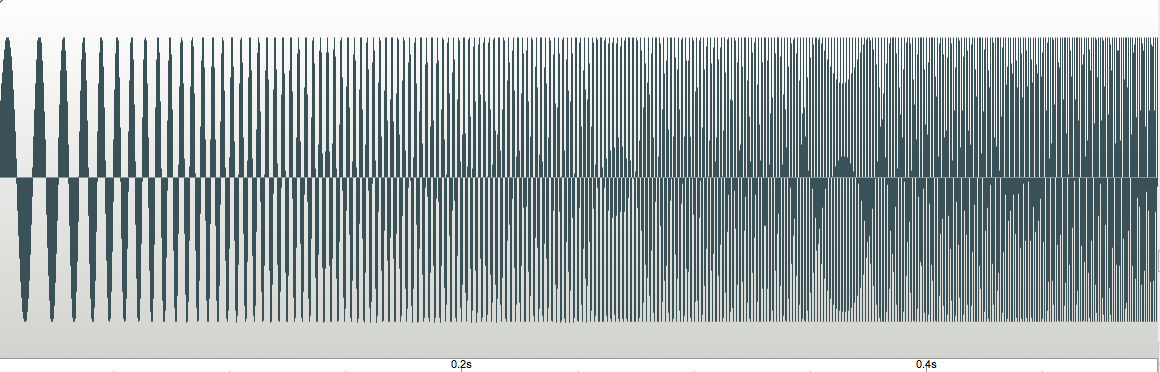


Fig 2(c)

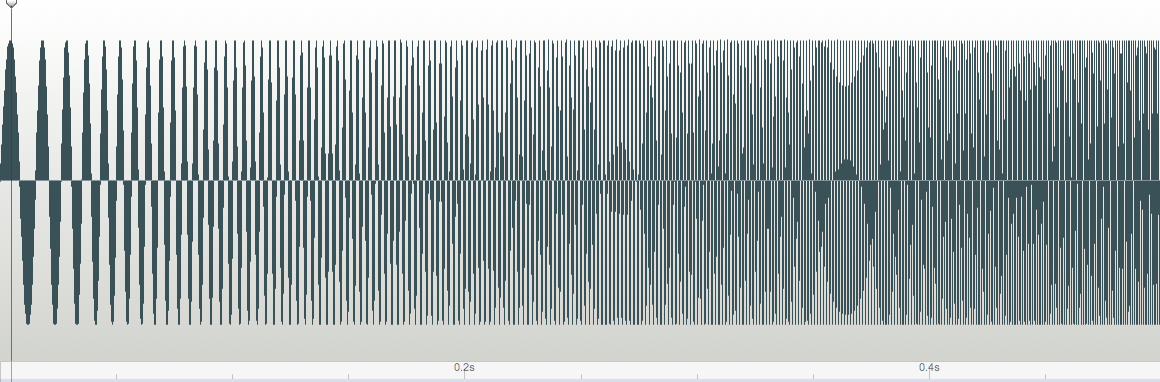


Fig 2(c)

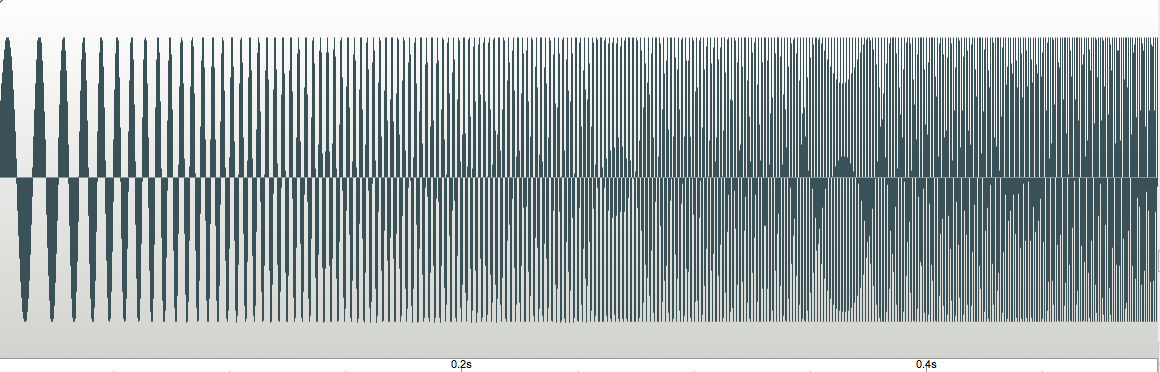


Fig 2(c)

5.1.2 Result and analysis on Audio4.wav

A sample Audio.wav is taken and all the above mention data hiding technique is used .

Fig 1(a) is the original audio1.wav cover file used for experiment.

Fig 1(b) is the reconstructed audio1.wav after embedding data in 2 bit planes using bit plane decomposition by catalan number system.

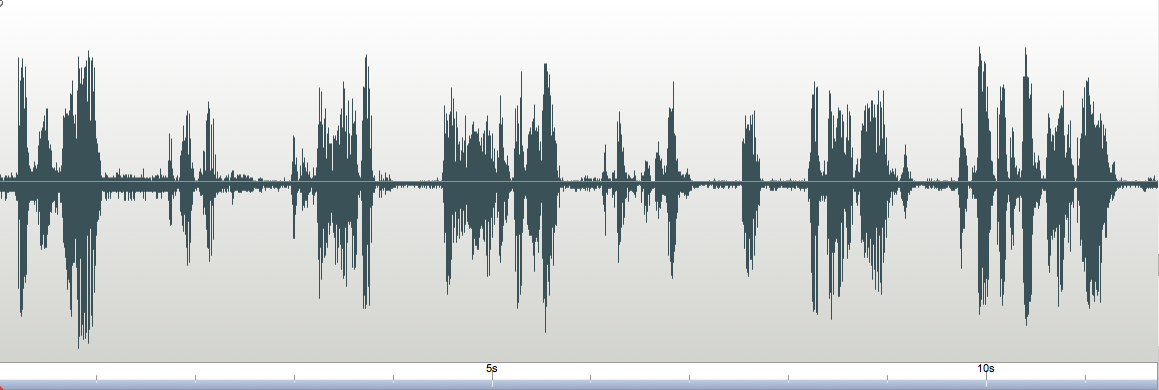
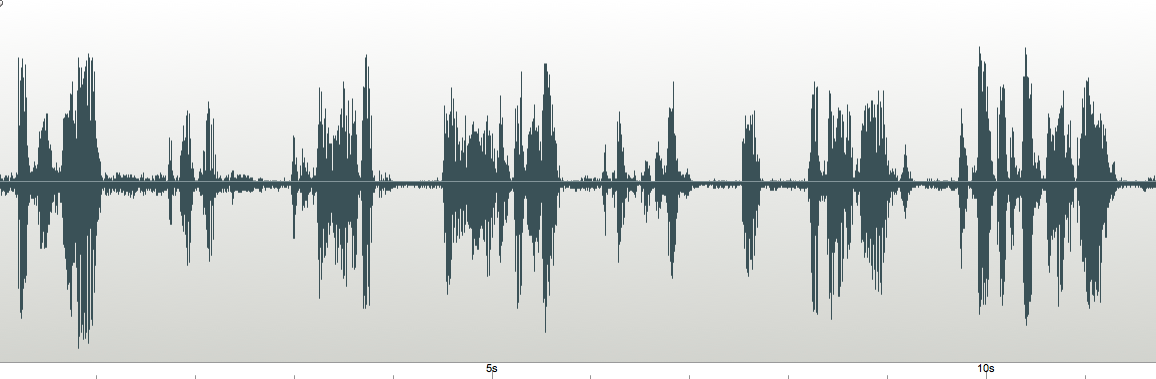
Fig 1(c) is the reconstructed audio1.wav after embedding data in 2 bit planes using bit plane decomposition by prime number system.

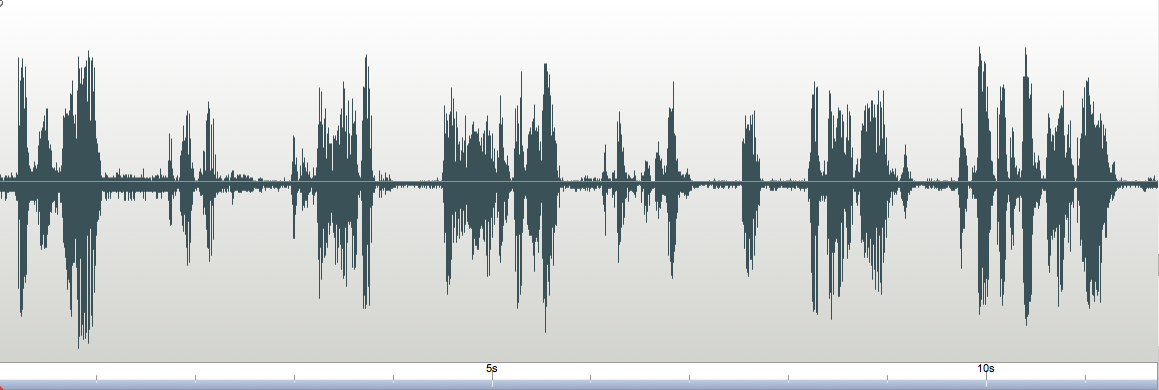
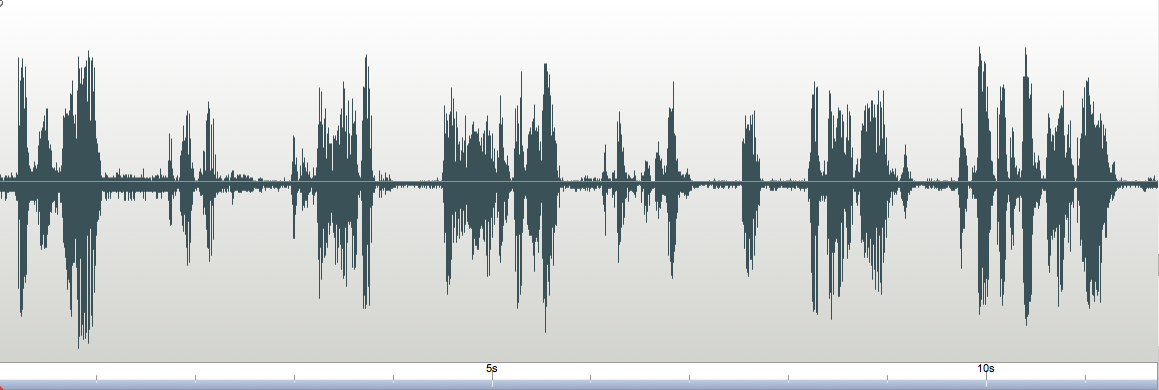
| Table 5: Results for audio4.wav | | | | |
| --- | --- | --- | --- | --- |
|  | SNR | total sample | used | spcc |
| prime | 29.5849 | 19802 | 8474 | 0.99994 |
| catalan | 36.603 | 14467 | 8474 | 0.99988 |
| fibonacci | 34.8938 | 13616 | 8474 | 0.99976 |
| fib-(2) | 25.1740 | 31632 | 8474 | 0.99966 |
| fib(1,2) | 29.940 | 8474 | 8474 | 0.99954 |

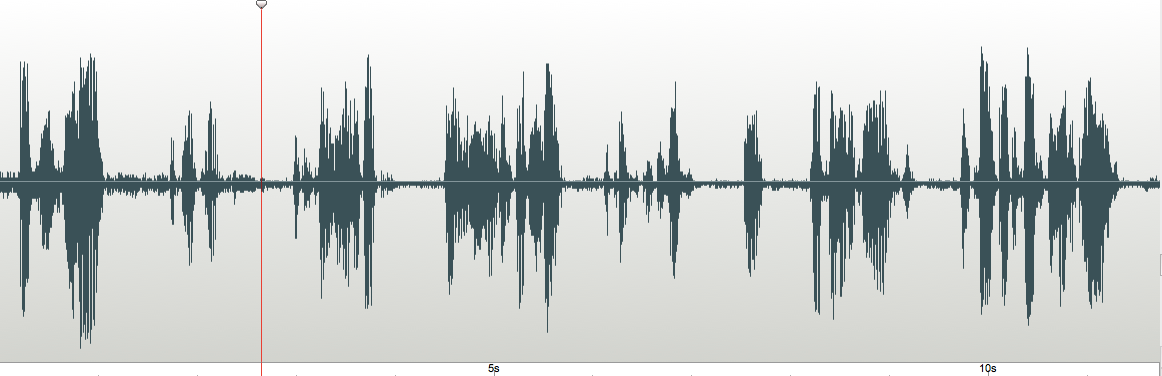
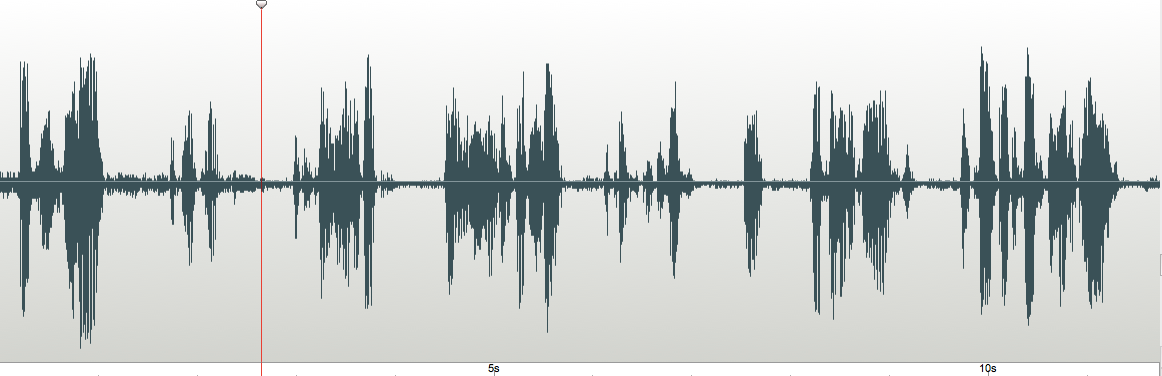
Fig 1(d) is the reconstructed audio1.wav after embedding data in 2 bit planes using bit plane decomposition by fibonacci number system.

Fig 1(e) is the reconstructed audio1.wav after embedding data in 2 bit planes using bit plane decomposition by fibonacci(2) system.

Fig 1(f) is the reconstructed audio1.wav after embedding data in 2 bit planes using bit plane decomposition by fibonacci(1,2) system.







References

[1.] F. Battisti, M. Carli, A. Neri, K. Egiaziarian, “A Generalized Fibonacci LSB Data Hiding Technique, 3rd International Conference on Computers and Devices for Communication (CODEC‐06), Institute of Radio Physics and Electronics, University of Calcutta, December 18‐20, 2006.

[2.] Chi‐Kwong Chan and L. M. Cheng, “Hiding data in images by simple LSB substitution,” Pattern Recognition, pp. 469–474, Mar. 2004.

[3.] Walter Bender, Daniel Gruhl, Norishighe Morimoto, and A. Lu, “Techniques for data hiding." I.B.M. Systems Journal, vol. 35, no. 3 & 4, pp. 313{336, 1996}.

[4.] Sandipan Dey, Ajith Abraham and Sugata Sanyal "An LSB Data Hiding Technique Using Prime Numbers", Third International Symposium on Information Assurance and Security, August 29‐31, 2007, Manchester, United Kingdom, IEEE Computer Society press, USA, ISBN 0‐7695‐2876‐7, pp. 101‐106, 2007.

[5.]Steganographic Techniques and their use in an Open Systems Environment‐Bret Dunbar, The Information Security Reading Room, SANS Institute 2002.

# [6.]Data Hiding Techniques Using Prime and Natural Numbers [Sandipan Dey](https://arxiv.org/search/cs?searchtype=author&query=Dey%2525252C+S),[Ajith Abraham](https://arxiv.org/search/cs?searchtype=author&query=Abraham%2525252C+A),[Bijoy Bandyopadhyay](https://arxiv.org/search/cs?searchtype=author&query=Bandyopadhyay%2525252C+B),[Sugata Sanyal](https://arxiv.org/search/cs?searchtype=author&query=Sanyal%2525252C+S), Journal of Digital Information Management, ISSN 0972-7272, Volume 6, No 3, pp. 463-485, 2008.

[7.]Substitutive steganography in the generalized Fibonacci domain,Mammi. E,Battisti, F, Carli. M, Neri. A, Egiazarian. K Proceedings of the SPIE, Volume 7245, id. 72450P (2009),02/2009

[8.]On Embedding of Text in Audio A Case of Steganography, Pramatha Nath Basu, Tanmay Bhowmik,2010 International Conference on Recent Trends in Information, Telecommunication and Computing

[9.]Data Hiding Techniques in Steganography Using Fibonacci and Catalan Numbers,Nikolaos Aroukatos, Kostas Manes, Stelios Zimeras, Fotis Georgiakodis, 2012 Ninth International Conference on Information Technology - New Generations