**PROJECT CYMPHONY**

**A PROJECT REPORT**

###### ***Submitted by***

##### **P**ratul Maurya(19BCY10036)

**Akshat Verma** (19BCY10075)

**Param Chawla** (19BCY10117)

*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF TECHNOLOGY**

*in*

# **COMPUTER SCIENCE AND ENGINEERING (Speci**alization in Cyber Security and Digital Forensics)



**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING**

**VIT BHOPAL UNIVERSITY**

**KOTHRI KALAN, SEHORE**

**MADHYA PRADESH - 466114**

##### NOV 2020

**VIT BHOPAL UNIVERSITY,KOTHRI KALAN, SEHORE**

**MADHYA PRADESH – 466114**

**BONAFIDE CERTIFICATE**

Certified that this project report titled **PROJECT CYMPHONY** is the bonafide work of “**PRATUL MAURYA (19BCY10036), AKSHAT VERMA (19BCY10075), PARAM CHAWLA (19BCY10117)** who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported here does not form part of any other project / research work on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

**PROGRAM CHAIR PROJECT GUIDE**

Dr. Shishir K. Shandilya Dr. Praveen Lalwani

(Division Head Cyber Security ​& Digital Forensics​) (Assistant Professor)

School of Computer Science and Engineering School of Computer Science and Engineering

VIT BHOPAL UNIVERSITY VIT BHOPAL UNIVERSITY

The Project Exhibition I Examination is held on 28 October 2020.

**ACKNOWLEDGEMENT**

First and foremost I would like to thank the Lord Almighty for His presence and immense blessings throughout the project work.

I wish to express my heartfelt gratitude to Dr. Shishir K. Shandilya, Head of the Department, School of Computer Science and Engineering (Cyber Division), for much of his valuable support and encouragement in carrying out this work.

I would like to thank my internal guide Dr. Praveen Lalwani,for continually guiding and actively participating in my project, giving valuable suggestions to complete the project work.

I would like to thank all the technical and teaching staff of the School of Computer Science and Engineering, who extended directly or indirectly all support.

Last, but not the least, I am deeply indebted to my parents who have been the greatest support while I worked day and night for the project to make it a success.

**LIST OF ABBREVIATIONS**

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **TITLE** | **PAGE NO.** |
|  |  |  |
|  |  |  |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **TABLE NO.** | **TITLE** | **PAGE NO.** |
| **1** |  |  |

**ABSTRACT**

Steganography has been proposed as a new alternative technique to enforce data security. Lately, novel and versatile audio steganographic methods have been proposed. A perfect audio Steganographic technique aims at embedding data in an imperceptible, robust and secure way and then extracting it by authorized people. Hence, up to date the main challenge in digital audio steganography is to obtain robust high capacity steganographic systems. Leaning towards designing a system that ensures high capacity or robustness and security of embedded data has led to great diversity in the existing steganographic techniques. In this project, we present a current state of art literature in digital audio steganographic techniques. We explore their potentials and limitations to ensure secure communication. A comparison and an evaluation for the reviewed techniques is also presented in this project.

In order to improve the data hiding in all types of multimedia data formats such as image and audio and to make hidden messages imperceptible, a novel method for steganography is introduced in this Project. It is based on Least Significant Bit (LSB) manipulation and inclusion of redundant noise as a secret key in the message. This method is applied to data hiding in images.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | List of Abbreviations  List of Figures  List of Tables  Abstract | 3  4  5  6 |
| 1 | **INTRODUCTION** Introduction 1.2 Motivation for the work  1.3 Evaluation by Steganalysis  1.4 Problem Statement  1.5 Objective of the work  1.6 Organization of the thesis  1.7 Summary | 9  9  10  10  10  11  11 |
| 2 | **LITERATURE SURVEY**  2.1 Introduction  2.2 Steganography  2.3 Existing Algorithms  2.3.1 Phase Coding  2.3.2 Frequency Modulation  2.3.3 Least Significant Bit  2.4 Modified LSB Algorithm  2.5 Research issues/observations from  literature Survey  2.6 Summary | 12  12  12  12  13  14  14  14  15 |
| 3 | **SYSTEM ANALYSIS**  3.1 Introduction  3.2 Disadvantages/Limitations in the existing system  3.3 Proposed System  3.4 Summary | 16  16  16  17 |
| 4 | **SYSTEM DESIGN AND IMPLEMENTATION**  4.1 Introduction  4.2 Encryption Process implementation  4.3 Decryption Process implementation  4.4 Summary | 18  18  19  19 |
| 5 | **PERFORMANCE ANALYSIS**  5.1Introduction  5.2 Performance Measures (Table/text)  5.3 Performance Analysis(Graphs/Charts)  5.4 Summary | 20  20  21  22 |
| 6 | **FUTURE ENHANCEMENT AND CONCLUSION**  6.1Introduction  6.2 Limitation/Constraints of the System  6.3 Future Enhancements  6.4 Conclusion | 23  23  23  23 |
|  | Appendices  References | 25  30 |

1. **INTRODUCTION**

**1.1 Introduction**

The word “Steganography”, like many other cool terms, has a Greek origin. It is derived from two Greek words steganos, meaning “covered,” and graphein, meaning “to write”, and refers to the art of enabling covert communication that uses clever methods to hide information in plain sight.

Note that Steganography is not the same as Encryption / Cryptography. In Steganography we do not aesthetically alter the publicly communicated message, meaning that the intelligibility of the “carrier” message remains intact. This has an important advantage that a “public” message doesn’t attract undue attention to itself, and the fact that this message is carrying a “secret” message is not known to eavesdroppers. On the other hand, encrypted messages are masked using well-known algorithms and arouse the curiosity of hackers and/or researchers to try and break the encryption.

**1.2 Motivation for the work**

The primary goal of steganography is to reliably send hidden information secretly, not merely to obscure its presence. Steganography in today’s computer era is considered a sub-discipline of the data communication security domain. Lately, new directions based on steganographic approaches started to emerge to ensure data secrecy. Rather than as a substitute to existing solutions, these approaches could achieve better data secrecy if combined with conventional security techniques. Modern techniques of steganography exploit the characteristics of digital media by utilizing them as carriers (covers) to hold hidden information. Covers can be of different types including image, audio, video, text, and IP datagram.

**1.3 Evaluation by Steganalysis**

The primary goal of steganography is to reliably send hidden information secretly, not merely to obscure its presence. Modern techniques of steganography exploit the characteristics of digital media by utilizing them as carriers (covers) to hold hidden information. Covers can be of different types including image, audio, video, text, and IP datagram. The sender embeds data of any type in a digital cover file using a key to produce a stego-file, in such a way that an observer cannot detect the existence of the hidden message. At the other end, the receiver processes the received stego-file to extract the hidden message. An obvious application of such steganographic systems is a covert communication using innocuous cover audio signals, such as telephone or video conference conversations.

**1.4 Problem Statement**

Our Project intends to overcome the existing problems and limitations of current Steganography techniques. We implement the LSB (Least Significant Bit) Algorithm for the same.

**1.5 Objective of the work**

Manipulation of sensitive images is a very important issue in modern transmission and storage. The objective of this study is to achieve the protection by applying a modified steganography method. In our project, We primarily concentrated on the data security issues when sending the data over the network using steganographic techniques.The main objectives of the project are -

• Requirement of this steganography system is that the secret message carried by stego-media should not be sensible to human beings.

• The other goal of steganography is to avoid drawing suspicion to the existence of a hidden message

• This approach of information hiding technique has recently become important in a number of application areas.

**1.6 Organization of the thesis**

1. INTRODUCTION
2. LITERATURE SURVEY
3. SYSTEM ANALYSIS
4. SYSTEM DESIGN AND IMPLEMENTATION
5. PERFORMANCE ANALYSIS
6. FUTURE ENHANCEMENT AND CONCLUSION
7. REFERENCES

**1.7 Summary**

The primary goal of steganography is to reliably send hidden information secretly, not merely to obscure its presence.

Modern techniques of steganography exploit the characteristics of digital media by utilizing them as carriers (covers) to hold hidden information.

Covers can be of different types including image, audio, video, text, and IP datagram.

The sender embeds data of any type in a digital cover file using a key to produce a stego-file, in such a way that an observer cannot detect the existence of the hidden message.

At the other end, the receiver processes the received stego-file to extract the hidden message. An obvious application of such a steganographic system is a covert communication using innocuous cover audio signals, such as telephone or video conference conversations.

1. **LITERATURE SURVEY**

**2.1 Introduction**

Electronic communication is the lifeblood of many organizations. Much of the information communicated on a daily basis must be kept conﬁdential. Informa- tion such as ﬁnancial reports, employee data and medical records needs to be communicated in a way that ensures conﬁdentiality and integrity. This makes good business sense and may even be regulated by legislation like the Health Insurance Portability and Accountability Act (HIPAA). The problem of insecure communication is compounded by the fact that much of this information is sent over the public Internet and may be processed by third parties, as in e-mail or instant messaging (IM).

**2.2 Steganography**

Steganography aims to hide information in a cover data in such a way that non-participating persons are not able to detect the presence of this information by analyzing the information detection. Unlike watermarking, steganography does not intend to prevent the hidden in- formation by opponents of removing or changing the hidden message, which is embedded in the cover data but it emphasizes on remaining undetectable. Steganography is particularly interesting for applications in which the encryption can not be used to protect the communication of conﬁdential information.

**2.3 Existing Algorithms**

There are numerous methods used to hide information inside of Audio files. The three most common methods are phase coding, frequency modulation and LSB.

**2.3.1 Phase coding**

Phase coding substitutes the phase of an initial audio segment with a reference phase that represents the hidden data. This can be thought of as sort of an encryption for the audio signal by using what is known as Discrete Fourier Transform (DFT), which is nothing more than a transformation algorithm for the audio signal. The Human Auditory System (HAS) can’t recognize the phase change in an audio signal as easily as it can recognize noise in the signal. The phase coding method exploits this fact. This technique encodes the secret message bits as phase shifts in the phase spectrum of a digital signal, achieving an inaudible encoding in terms of signal-to- noise ratio.

One disadvantage associated with phase coding is a low data transmission rate due to the fact that the secret message is encoded in the first signal segment only. This might be addressed by increasing the length of the signal segment. However, this would change phase relations between each frequency component of the segment more drastically, making the encoding easier to detect. As a result, the phase coding method is used when only a small amount of data, such as a watermark, needs to be concealed.

**2.3.2 Frequency Modulation**

The Frequency-modulation based method that we are going to discuss here is very powerful and doesn’t induce noise in the carrier signal. Using this method we will embed one “secret” audio onto another “public” audio. The secret audio shall be imperceptible to the human ear and the receiver would be able to extract the secret audio at their end.

The main idea of this algorithm is that we will conceal our secret audio in the near-ultrasound range while keeping our public audio data in the normal hearing range.

The sound quality is a bit poor, because for a sample rate of 44100 Hz and a carrier frequency of 17500 Hz, the audio bandwidth is less than 4500 Hz, which is not enough for high-quality audio. Hence, you should expect some distorted speech signal, because there is not much “room” for sound quality in the range between 17500 and 22050 Hertz on a soundcard with only 44100 Hertz sample frequency.

**2.3.3 Least Significant Bit**

Least significant bit (LSB) coding is the simplest way to embed information in a digital audio file. By substituting the least significant bit of each sampling point with a binary message, LSB coding allows for a large amount of data to be encoded. Among many different data hiding techniques proposed to embed secret messages within audio files, the LSB data hiding technique is one of the simplest methods for inserting data into digital signals in noise free environments, which merely embeds secret message-bits in a subset of the LSB planes of the audio stream.

This proposed system is to provide a good, efficient method for hiding the data from hackers and sent to the destination in a safe manner. This proposed system will not change the size of the file even after encoding and also suitable for any type of audio file format.

**2.4 Modified LSB**

In the modified LSB encoding technique we have seen that the 4th bit is set according to the secret message. If the sample bit is not equal to the secret message bit we then simply flip the rest of the bits of that given sample. In our proposed model we take consecutive two bits from the secret message and instead of changing a single bit in a sample we change two bits (4th and 3rd position) of the sample. If there is change in these two bits we flip the rest of the LSB otherwise there is no change. For example, if the original sample value was (0...01000)2=(8)10, and the watermark bits 01 are to be embedded into 4th and 3rd LSB layer, the standard algorithm will produce the value (0...00000)2=(0)10 to embed the 1st watermark bit only and for the 2nd bit we need another sample, the modified algorithm produces sample that has value (0...00111)2=(7)10, which is far more closer to the original one but here also we need another sample to embed the 2nd watermark bit.

**2.5 Research issues/observations from literature Survey**

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. Subjective listening test showed that, on average, the maximum LSB depth that can be used for LSB based watermarking without causing noticeable perceptual distortion is the fourth LSB layer when 16 bits per sample audio sequences are used. The tests were performed with a large collection of audio samples and individuals with different backgrounds and musical experience. None of the tested audio sequences had perceptual artifacts when the fourth LSB has been used for data hiding, although in certain music styles, the limit is even higher than the fourth LSB layer. Robustness of the watermark, embedded 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 fourth LSB layer for the standard LSB coding algorithm.

**2.6 Summary**

In this project we have introduced a robust method of imperceptible hiding data in an audio file. This system is to provide a good, efficient method for hiding the data from hackers and sent to the destination in a safe manner. This proposed system will not change the size of the file even after encoding and also suitable for any type of audio file format. Thus we conclude that audio data hiding techniques can be used for a number of purposes other than covert communication or deniable data storage , information tracing and fingerprinting, tamper detection. As the sky is not limited so is not for development. Man is now pushing away its own boundaries to make every thought possible. So similarly these operations described above can be further modified as it is in the world of Information Technology.

1. **SYSTEM ANALYSIS**

**3.1 Introduction**

We present an intelligent framework for steganography using LSB encoding for audio data. The idea of the algorithm is that two watermark bits are embedded which provides the minimal distortion of the fixed length host audio with high capacity.

The improvement in robustness in presence of additive noise is obvious, as the proposed algorithm obtains significantly lower bit error rates than the standard algorithm .

**3.2 Disadvantages/Limitations in the existing system**

Most of the projects and research paper that we found on audio steganography are limited only to a few bits of secret file also most of them distorted the public carrier audio file so much that after the steganography any eavesdropper could understand that something has been done to the audio file because of the distortions.

We proposed to increase the file size while keeping the audio quality of the public carrier file as similar as possible to the original carrier files characteristics audio characteristics

Its main drawback is that the output audio file that is to be sent to the receiver is highly distorted so it doesn’t really fulfill the main motive of audio steganography.

**3.3 Proposed System**

Our project intends to overcome the existing problems and limitations of current steganography programs.Our project is based on the Least Significant BIt Steganography.

This will allow us to create an output audio file in such a way that it retains the basic acoustic characteristics of the original carrier file.We proposed to increase the file size while keeping the audio quality of the public carrier file as similar as possible to the original carrier files characteristics audio characteristics

**3.4 Summary**

In this project we are using an audio file as a public carrier file where we are hiding a secret text message. The advantage of which is that we are able to exploit the characteristics of the digital media that is the audio file by utilising them as carriers to hold hidden information. The primary hole is to reliably send hidden information secretly. The problems to overcome were that the output of the carrier audio file was becoming distorted to some extent if we tried to use too many bits. And also to keep the audio characteristics as similar as possible to the original file we have to compromise with the size of the secret file message.

1. **WORK DONE**

**4.1 Introduction**

* The major goal of Steganography is to transmit data over a confidential media. Based on the type and the amount of data that needs to be transmitted carrier audio is selected.
* Audio steganography is a good technique for storing small volumes of data. In any steganographic technique there are two different phases. One is at the sender phase and the other is the receiver phase.
* In the sender side, an embedding algorithm is designed to embed the data into the audio file and at the receiver side retrieval algorithm is designed to retrieve the information.
* These two (Embedding and Retrieval) algorithms are implemented in a single program.
* Listening tests showed that the described algorithm succeeds in taking two bit positions to embed secret data without affecting the perceptual transparency of the carrier audio signal.
* The tests were performed with a large collection of audio samples and individuals with different backgrounds and musical experience.
* None of the tested audio sequences had perceptual artifacts when used for data hiding.

**4.2 Encryption Process implementation**

Step 1: Read the cover audio and text message which is to be hidden in the cover audio.

Step 2: Convert original audio stream into binary stream

Step 3: Convert secret message into binary stream

Step 4: Calculate LSBs of each bit of carrier audio.

Step 5: Replace LSBs of cover audio with each bit of secret message sequentially.

Step 6: Repeat step 6 until the entire secret bits are embedded. Then place the binary representation of the target character at the end of the embedding.

Step 7: Write stego audio

**4.3 Decryption Process implementation**

Step 1: Read the stego audio.

Step 2: Convert stego audio stream into binary stream

Step 3: Calculate LSBs of each bit of stego audio.

Step 4: Retrieve bits & convert each 8 bit into character.

Step 5: Repeat Step 2 and 3 until the binary representation of the target character is found.

Step 6: The retrieved characters are placed sequentially to get back the original secret message.

**4.4 Summary**

The user has to provide the program with two files i.e. the secret text message and the public carrier sound that will be used to hide the message. The user then chooses the number of Least Significant Bits to use and then the program embeds the secret data with the public carrier sound to give an output sound file. On the receiver’s end the user has to provide the embedded sound file as input along with the number of Least Significant Bits to use and the bytes of data hidden. Then the program gives the secret message from the embedded audio as a text file output. However it must be noted that entering even one of them wrong (LSBs used or bytes of data hidden) will result in getting a corrupted text output or an incomplete one.

1. **OBSERVATION**

**5.1 Introduction**

Here we would like to give our observations over our entire project.

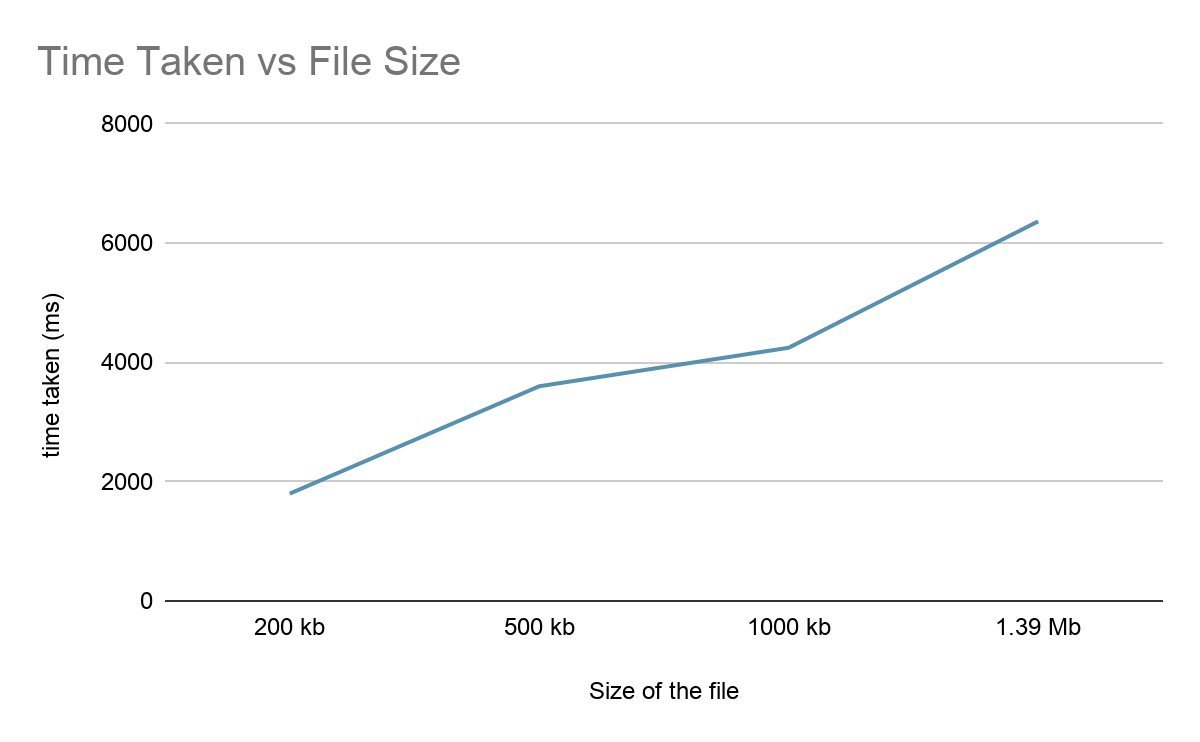
**5.2 Performance Measures (Table/text)**

Our code has been used to perform steganography with various kinds of files with varying sizes. The maximum being : 200KB and the minimum being: 1.34MB and is performed in the following manner

|  |  |  |
| --- | --- | --- |
| File size | file type | Mode |
| 200KB | Audio | LSB |
| 500KB | Audio | LSB |
| 1000KB | Audio | LSB |
| 1.34MB | Audio | LSB |

In this table we can see all the files taken for conducting this test and noticing the behaviour of the code with respect to taken in conducting the whole process

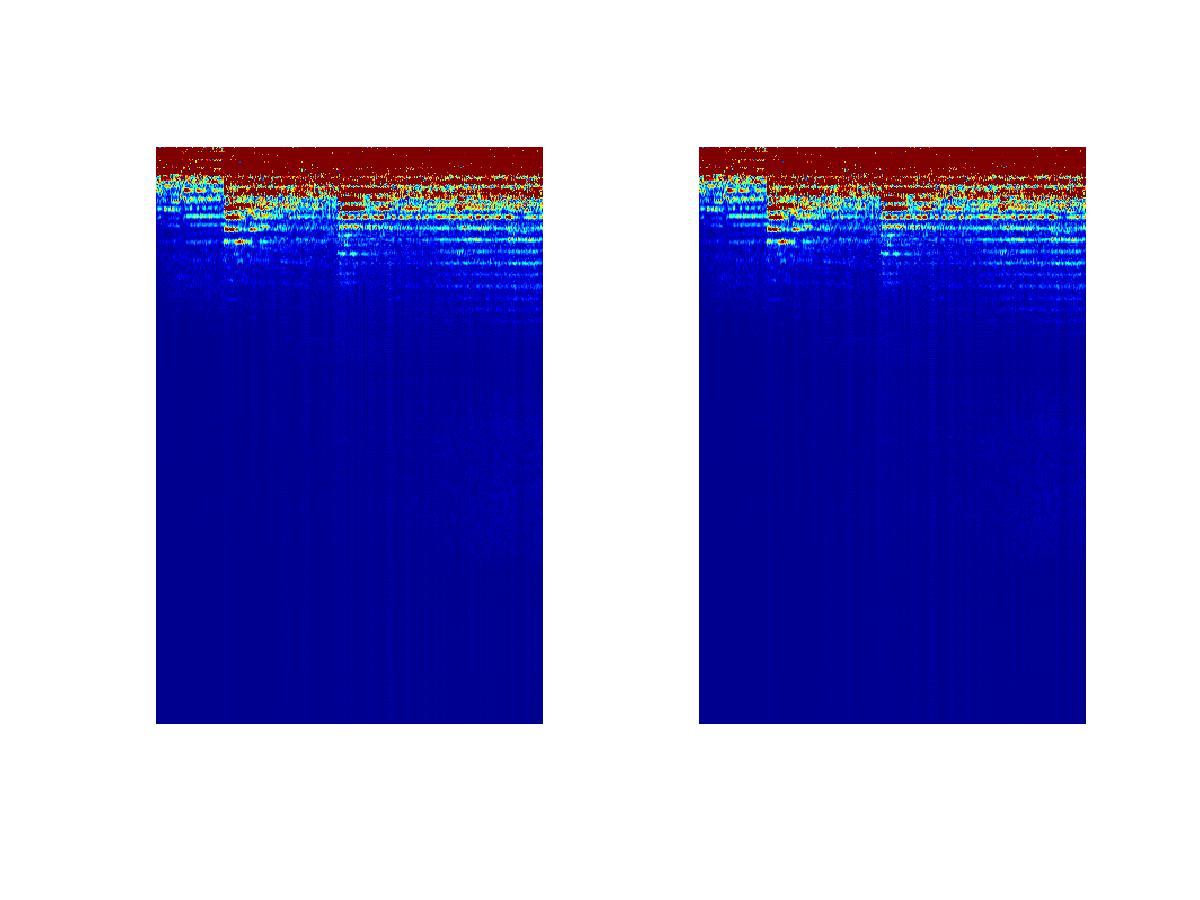
**5.3 Performance Analysis(Graphs/Charts)**

In the following graph we can see clearly how much time it takes with respect to the file size taken to embed the text files in the audio file.We can sum this whole graph very easily as to comment about how time taken is proportionate to the file size.

**Quality of the Final file vs Encrypted file**

When we ran the originality test on both the sounds before and after embedding it with the text file.(Website used for the test [blue2digital](https://blue2digital.com/apps/compare-audios.html) )

**The similarity between these two audio files is 100.00%.**

****

*\* The image above displays the spectrograms of the two uploaded audio files*

**5.4 Summary**

By the above result we can conclude that this code can encrypt and do steganography with much greater efficiency than expected.

1. **RECOMMENDATION FOR FUTURE WORK**

**6.1 Introduction**

We would like to state that these improvements couldn’t be done on this project because of a limited time frame and not having prior knowledge of the implementation of the working model before.

All of these improvements are totally based upon all the extra material that can be attached to the code and to increase the multitasking index of the specific code provided by us.

**6.2 Limitation/Constraints of the System**

Currently, this method can only be used for small text files(not more than: 500KB)

This system has not been used in a working GUI. Which makes it difficult to use for a layman.

This software is still limited to text files and cannot be used to encrypt video and image files.

**6.3 Future Enhancements**

To ensure further security we intend to implement encryption along with steganography. A stream cipher to encrypt the data before embedding it into the carrier audio file will ensure that the data is not in plain text format. Also the secret message is still limited to text files although large in size but in future we would like to embed photos and other media files as well.

**6.4 Conclusion**

In order to provide better protection to digital data content, new steganography techniques have been investigated in recent researcher works. The availability and popularity of digital audio signals have made them an appealing choice to convey secret information. Audio steganography techniques address issues related to the need to secure and preserve the integrity of data hidden in voice communications in particular. In this work, a comparative study of the current-state-of-the-art literature in digital audio steganography techniques and approaches is presented. In an attempt to reveal their capabilities in ensuring secure communications, we discussed their strengths and weaknesses. Also, a differentiation between the reviewed techniques based on the intended applications has been highlighted. Thus, while temporal domain techniques, in general, aim to maximize the hiding capacity, transform domain methods exploit the masking properties in order to make the noise generated by embedded data imperceptible. On the other side, encoded domain methods strive to ensure the integrity of hidden data against challenging environments such as real time applications. To better estimate the robustness of the presented techniques, a classification based on their occurrence in the voice encoder is given. A comparison as well as a performance evaluation (i.e., imperceptibility and steganalysis) for the reviewed techniques have been also presented. This study showed that the frequency domain is preferred over the temporal domain and music signals are better covers for data hiding in terms of capacity, imperceptibility and undetectability. From our point of view, the diversity and large number of existing audio steganography techniques expand application possibilities. The advantage of using one technique over another one depends on the application constraints in use and its requirement for hiding capacity, embedded data security level and encountered attacks resistance.

**APPENDICES**

|  |
| --- |
| **import** **getopt**, **os**, **sys**, **math**, **struct**, **wave**  **def** print\_usage():  **print**("**\n**Usage options:**\n**",  "-h, --hide The script runs to hide data**\n**",  "-r, --recover The script runs to recover data**\n**",  "-s, --sound Write name of carrier wav file**\n**",  "-d, --data Write file name having data to hide**\n**",  "-o, --output Output filename of your choice**\n**",  "-n, --nlsb Number of LSBs you want to use**\n**",  "-b, --bytes Number of bytes to recover from sound file**\n**"  " --help Display help**\n**")  **def** prepare(sound\_path):  **global** sound, params, n\_frames, n\_samples, fmt, mask, smallest\_byte  sound = wave.open(sound\_path, "r")    params = sound.getparams()  num\_channels = sound.getnchannels() *#Returns number of audio channels (1 for mono, 2 for stereo).*  sample\_width = sound.getsampwidth() *#Returns sample width in bytes.*  n\_frames = sound.getnframes() *#Returns number of audio frames.*  n\_samples = n\_frames \* num\_channels  **if** (sample\_width == 1): *# samples are unsigned 8-bit integers*  fmt = "{}B".format(n\_samples)  *# Used to set the least significant num\_lsb bits of an integer to zero*  mask = (1 << 8) - (1 << num\_lsb)  *# The least possible value for a sample in the sound file is actually*  *# zero, but we don't skip any samples for 8 bit depth wav files.*  smallest\_byte = -(1 << 8)  **elif** (sample\_width == 2): *# samples are signed 16-bit integers*  fmt = "{}h".format(n\_samples)  *# Used to set the least significant num\_lsb bits of an integer to zero*  mask = (1 << 15) - (1 << num\_lsb)  *# The least possible value for a sample in the sound file*  smallest\_byte = -(1 << 15)  **else**:  *# Python's wave module doesn't support higher sample widths from some reason*  **raise** **ValueError**("File has an unsupported bit-depth")  **def** hide\_data(sound\_path, file\_path, output\_path, num\_lsb):  **global** sound, params, n\_frames, n\_samples, fmt, mask, smallest\_byte  prepare(sound\_path)  *# We can hide up to num\_lsb bits in each sample of the sound file*  max\_bytes\_to\_hide = (n\_samples \* num\_lsb) // 8  filesize = os.stat(file\_path).st\_size    **if** (filesize > max\_bytes\_to\_hide):  required\_LSBs = math.ceil(filesize \* 8 / n\_samples)  **raise** **ValueError**("Input file too large to hide, "  "requires {} LSBs, using {}"  .format(required\_LSBs, num\_lsb))    **print**("Using {} B out of {} B".format(filesize, max\_bytes\_to\_hide))      *# Put all the samples from the sound file into a list*  raw\_data = list(struct.unpack(fmt, sound.readframes(n\_frames)))  sound.close()    input\_data = memoryview(open(file\_path, "rb").read())    *# The number of bits we've processed from the input file*  data\_index = 0  sound\_index = 0    *# values will hold the altered sound data*  values = []  buffer = 0  buffer\_length = 0  done = False    **while**(**not** done):  **while** (buffer\_length < num\_lsb **and** data\_index // 8 < len(input\_data)):  *# If we don't have enough data in the buffer, add the*  *# rest of the next byte from the file to it.*  buffer += (input\_data[data\_index // 8] >> (data\_index % 8)  ) << buffer\_length  bits\_added = 8 - (data\_index % 8)  buffer\_length += bits\_added  data\_index += bits\_added    *# Retrieve the next num\_lsb bits from the buffer for use later*  current\_data = buffer % (1 << num\_lsb)  buffer >>= num\_lsb  buffer\_length -= num\_lsb  **while** (sound\_index < len(raw\_data) **and**  raw\_data[sound\_index] == smallest\_byte):  *# If the next sample from the sound file is the smallest possible*  *# value, we skip it. Changing the LSB of such a value could cause*  *# an overflow and drastically change the sample in the output.*  values.append(struct.pack(fmt[-1], raw\_data[sound\_index]))  sound\_index += 1  **if** (sound\_index < len(raw\_data)):  current\_sample = raw\_data[sound\_index]  sound\_index += 1  sign = 1  **if** (current\_sample < 0):  *# We alter the LSBs of the absolute value of the sample to*  *# avoid problems with two's complement. This also avoids*  *# changing a sample to the smallest possible value, which we*  *# would skip when attempting to recover data.*  current\_sample = -current\_sample  sign = -1  *# Bitwise AND with mask turns the num\_lsb least significant bits*  *# of current\_sample to zero. Bitwise OR with current\_data replaces*  *# these least significant bits with the next num\_lsb bits of data.*  altered\_sample = sign \* ((current\_sample & mask) | current\_data)  values.append(struct.pack(fmt[-1], altered\_sample))  **if** (data\_index // 8 >= len(input\_data) **and** buffer\_length <= 0):  done = True    **while**(sound\_index < len(raw\_data)):  *# At this point, there's no more data to hide. So we append the rest of*  *# the samples from the original sound file.*  values.append(struct.pack(fmt[-1], raw\_data[sound\_index]))  sound\_index += 1    sound\_steg = wave.open(output\_path, "w")  sound\_steg.setparams(params)  sound\_steg.writeframes(b"".join(values))  sound\_steg.close()  **print**("Data hidden over {} audio file".format(output\_path))  **def** recover\_data(sound\_path, output\_path, num\_lsb, bytes\_to\_recover):  *# Recover data from the file at sound\_path to the file at output\_path*  **global** sound, n\_frames, n\_samples, fmt, smallest\_byte  prepare(sound\_path)  *# Put all the samples from the sound file into a list*  raw\_data = list(struct.unpack(fmt, sound.readframes(n\_frames)))  *# Used to extract the least significant num\_lsb bits of an integer*  mask = (1 << num\_lsb) - 1  output\_file = open(output\_path, "wb+")    data = bytearray()  sound\_index = 0  buffer = 0  buffer\_length = 0  sound.close()  **while** (bytes\_to\_recover > 0):    next\_sample = raw\_data[sound\_index]  **if** (next\_sample != smallest\_byte):  *# Since we skipped samples with the minimum possible value when*  *# hiding data, we do the same here.*  buffer += (abs(next\_sample) & mask) << buffer\_length  buffer\_length += num\_lsb  sound\_index += 1    **while** (buffer\_length >= 8 **and** bytes\_to\_recover > 0):  *# If we have more than a byte in the buffer, add it to data*  *# and decrement the number of bytes left to recover.*  current\_data = buffer % (1 << 8)  buffer >>= 8  buffer\_length -= 8  data += struct.pack('1B', current\_data)  bytes\_to\_recover -= 1  output\_file.write(bytes(data))  output\_file.close()  **print**("Data recovered to {} text file".format(output\_path))  **try**:  opts, args = getopt.getopt(sys.argv[1:], 'hrs:d:o:n:b:',  ['hide', 'recover', 'sound=', 'data=',  'output=', 'nlsb=', 'bytes=', 'help'])  **except** getopt.GetoptError:  print\_usage()  sys.exit(1)  hiding\_data = False  recovering\_data = False  **for** opt, arg **in** opts:  **if** opt **in** ("-h", "--hide"):  hiding\_data = True  **elif** opt **in** ("-r", "--recover"):  recovering\_data = True  **elif** opt **in** ("-s", "--sound"):  sound\_path = arg  **elif** opt **in** ("-d", "--data"):  file\_path = arg  **elif** opt **in** ("-o", "--output"):  output\_path = arg  **elif** opt **in** ("-n", "--nlsb"):  num\_lsb = int(arg)  **elif** opt **in** ("-b", "--bytes"):  bytes\_to\_recover = int(arg)  **elif** opt **in** ("--help"):  print\_usage()  sys.exit(1)  **else**:  **print**("Invalid argument {}".format(opt))  **try**:  **if** (hiding\_data):  hide\_data(sound\_path, file\_path, output\_path, num\_lsb)  **if** (recovering\_data):  recover\_data(sound\_path, output\_path, num\_lsb, bytes\_to\_recover)  **except** **Exception** **as** e:  **print**("Ran into an error during execution. Check input and try again.**\n**")  **print**(e)  print\_usage()  sys.exit(1) |

**REFERENCES**

1. 1.M. L. Mat Kiah, B. B. Zaidan, A. A. Zaidan, A. Mohammed Ahmed and Sameer Hasan Al-bakri, "A review of audio based steganography and digital Watermarking", International Journal of the Physical Sciences, Vol. 6(16), pp. 3837-3850, 18 August, 2011.
2. Zamani, M., Manaf, A, Ahmad, R.B., Jaryani, F., Taherdoost H., Zeki, AM.,"A secure audio steganography approach", International Conference for Internet Technology and Secured Transactions, Page(s):1 - 6, 2009.
3. .Kaliappan Gopalan, "A Unified Audio and Image Steganography by Spectrum Modification", International Conference on Industrial Technology, Page(s):1 - 5, 2009.
4. Cvejic N. and Seppänen T. ”Increasing the capacity of LSB based audio steganography”, Proc. 5th IEEE International Workshop on Multimedia Signal Processing, St. Thomas, VI, December 2002, pp. 336- 338.
5. <https://alphamanual.audacityteam.org/>
6. <https://www.ijcsmc.com/docs/papers/April2014/V3I4201468.pdf>
7. <https://medium.com>
8. <https://docs.python.org/>
9. [https://asmp-eurasipjournals.springeropen.com/articles/10.1186/1687-4722-2012-25](https://asmp-eurasipjournals.springeropen.com/articles/10.1186/1687-4722-2012-25#author-information)
10. [https://www.ijirae.com/volumes/vol1/](https://www.ijirae.com/volumes/vol1/issue10/46.NVCS10104.pdf)