STEGANOGRAPHY TOOLS VISUALIZER

A PROJECT REPORT

Submitted by

SANJIPAN DEB	21BCY10015
ABANTEEKA ACHARYA	21BCY10005
PRIYANSHI PANCHAL	21BCY10145

in partial fulfillment for the award of degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

(Specialization in Cybersecurity and Digital Forensics)

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

VIT BHOPAL UNIVERSITY

KOTHRI KALAN, SEHORE

MADHYA PRADESH - 466114

JANUARY 2023

VIT BHOPAL UNIVERSITY, KOTHRI KALAN, SEHORE
MADHYA PRADESH – 466114

BONAFIDE CERTIFICATE

Certified that this project report titled **"STEGANOGRAPHY TOOLS VISUALIZER"** is the bonafide work of **"SANJIPAN DEB(21BCY10015), ABANTEEKA ACHARYA(21BCY10005), PRIYANSHI PANCHAL(21BCY10145)"** who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported at this time does not form part of any other project/research work based on which a degree or award was conferred on an earlier occasion to this or any other candidate.

PROGRAM CHAIR	**PROJECT SUPERVISOR**
Dr. R.Rakesh, Program Chair, Division of Cyber Security and Digital Forensics School of Computing Science and Engineering VIT BHOPAL UNIVERSITY	Dr. Hariharasitaraman S, Assistant Professor, Head-IQAC, School of Computing Science and Engineering VIT BHOPAL UNIVERSITY

The Project Exhibition II Examination is held on _______________

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. Pushpinder Singh Patheja**, Head of the Department, School of
Aeronautical Science for much of his valuable support and encouragement in carrying out this work.

I would like to thank my internal guide **Dr. Hariharasitaraman. S**, 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 Computing Science, who extended directly or indirectly all support.

Last, but not 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

- **LSB :** Least Significant Bit
- **Stego :** Steganography
- **RC4** : Rivest Cipher 4SM
- **KSA** : Key-Scheduling Algorithm
- **PRGA**: Pseudo random generation Algorithm (Stream Generation)
- **ZWC** : Zero Width Character
- **SM** : Secret Message
- **CT** : Cover Text
- **ASCII** : American Standard Code for Information Interchange
- **OS** : Operating System

LIST OF FIGURES AND GRAPHS

ABSTRACT

Steganography is the practice of representing information within another message or physical object, in such a manner that the presence of the information is not evident to human inspection. In computing contexts, a text message is concealed within another audio, video, text or image file. The word “Steganography” originates from the Greek word “steganographia” where “stegano” means “covered or concealed” and “graphy” means “writing”.

The advantages 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 many in themselves are 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 and its contents.

Through the concept of steganography, this project wishes to hide the text message in the cover files like audio, video, text, image files. The purpose in doing so is to create a fine multi purpose Steganography Tool to increase the security of confidential messages across a network when only sender and receiver knows the trick behind the steganographic message.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1 Introduction

Steganography is the art and science of writing hidden messages in such a way that no one apart from the sender and intended recipient, suspects the existence of the message, a form of security through obscurity. The word steganography is of Greek origin and means “concealed writing” from the Greek words steganos meaning “cover or protected”, and graphy meaning “writing”. The first recorded use of the term was in 1499 by Johannes Trithemius in his “Steganographia”, a treatise on cryptography and steganography disguised as a book on magic. Generally, messages will appear to be something else: image, articles, shopping lists or some other converted text and classically, the hiding message may be in invisible ink between the visible lines of a private letter.

The advantage of steganography over cryptography alone is that messages don't attract attention to themselves. Plainly visible encrypted messages, no matter how unbreakable, will arouse suspicion, and may in themselves be incriminating in countries where encryption is illegal. Therefore, whereas cryptography protects the contents of a message, steganography can be said to protect both messages and communicating parties.

Through the concept of steganography, this project wishes to hide the text message in the cover files like audio, video, text, image files. The purpose in doing so is to create a fine multi-purpose Steganography Tool to increase the security of confidential messages across a network when only sender and receiver knows the trick behind the steganographic message.

1.2 Problem Statement

As started earlier in the introduction, visible encryption messages will draw suspicion from others. This will make them attempt to decrypt the cipher text. Indirectly it raises the chance of a message being viewed by a third party with the exception of sender and viewer.

However, by hiding or embedding the text message into cover files like audio, video, text or image files, it is hard to arouse the suspicion of others whether it is containing any secret message or not as the processed file did not look differ from the original cover file in the naked eyes. This will make the message more secure. Indirectly, only the sender and the receiver know the threat behind the image and hence ensures that the message can only be viewed by sender and receiver.

1.3 Objective

The objectives of the project are to:

1. To store text messages into four types of cover files i.e.: Audio, Video, Text, Image.
2. To protect the text message from being viewed by third parties using encryption.
3. To restore the stored text message from the cover file.

1.4 Scope

The scopes involved in the project are steganography, cryptography, security issues and a slightly watermarking technique.

Steganography, as the introduction states, basically steganography was hiding something inside into another medium. This was exactly what will be done by us in this project as we will continue to attempt to hide the text file in image.

Cryptography involved the encryption and decryption process. We will try to encrypt by hiding the text message in audio, video, text and image files. The receiver shall be able to decrypt the steganographic message in order to read the original message which wishes to be sent by the sender.

The security issues involved here is what secures the steganographic message be. Is it really able to protect it from being viewed by a third party?

Watermarking technique shares some similar concepts with steganography as both wish to embed one thing going into another medium. Hence, we should use the help of watermarking techniques to embed text messages into image files and video frames. For text messages in text files we will use the Unicode method with specific zero-width characters and for audio we will use the least significant bit of the audio waves.

Basically, the overall process of this project can be viewed as below:

1. Choose the ideal cover file.

2. Embedded the text message taken from the user into the cover file.

3. Send the steganographic message to the receiver.

4. Receiver decrypt the steganographic message and restore back the message from the cover file.

1.5 Thesis Organization

This thesis consists of five (5) chapters. Chapter 1 will discuss the introduction to the system which will explain the introduction, problem statement, objective and scope. For Chapter 2, it will discuss the literature review, definition and types of steganography. For Chapter 3, it will discuss the methodology and software, hardware requirements of the project. For Chapter 4, it will discuss the design and implementation of the project. Chapter 5, it will show the results and discussion.

CHAPTER 2
LITERATURE REVIEW

2.1 Introduction

In this chapter, we are going to focus on discussing the results or findings based on the article, journals or any other related reference material. Some original words from the reference material may be cited in order to enhance the review. Purpose of this chapter is to explain about the selected project.

Basically, it is divided into a few sub-sections as well. Those sub-section include some little explanation of basic concepts of the selected project, research of some already existing similar problem or solution done by others and the hardware, technique or method which will be applied or used in the selected project.

This review will do research and describe the existing problem or solution done by other parties. It will also study other systems which are related to the selected project.

This chapter explains in detail regarding the techniques/method/hardware or technologies which are suitable to be adapted into the project. This chapter contains information about the study of the project in general.

2.2 What is Steganography?

Steganography is the art and science of writing hidden messages in such a way that no one apart from the sender and intended recipient, suspects the existence of the message, a form of security through obscurity. The word steganography is of Greek origin and means “concealed writing” from the Greek words steganos meaning “cover or protected”, and graphy meaning “writing”.

For the first time, this term was used in years 1499 when Johannes Trithemius is using it in his “Steganographia”, a dissertation on cryptography and steganography. In general, messages will be shown to be another thing else such as images, audio, video and text file.
Classically, this hidden message can be even in invisible ink between the visible lines of a

personal letter.

Steganography is hiding a message, rather than encoding it. If a message is not being suspected then it is quite hard to start to decode or decrypt it. It includes a vast array of skill and techniques for hiding messages in a variety of media.

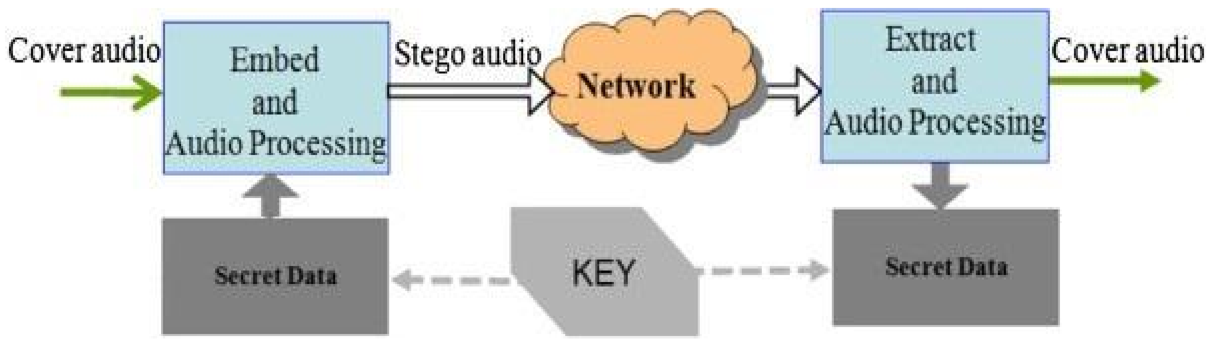
The benefit of steganography is that it does not draw attention to itself. A plainly visible an encrypted message will raise doubt no matter how unbreakable it was. Sometimes, the encrypted message using cryptography may be incriminating in some countries where encryption is considered illegal. Hence, whereas cryptography protects the contents of a message, steganography can be said to protect both communicating parties and messages

Using the theory of steganography, we were wishing to hide a text message in the four types of cover files i.e.: Audio,Video,Text,Image. The encrypted cover file will look like it does not differ from the original cover file. This can eliminate the suspicion of third parties toward the encrypted cover file. The text message embedded in the cover file should be able to retrieve it back to the original state.

2.3 Audio Steganography

Audio steganography is about hiding the secret message into the audio. It is a technique used to secure the transmission of secret information or hide their existence in an audio file. It also may provide confidentiality to secret message if the message is encrypted

Flowchart For the algorithm Procedure:



Wave module : We will use the wave module to read the audio file.

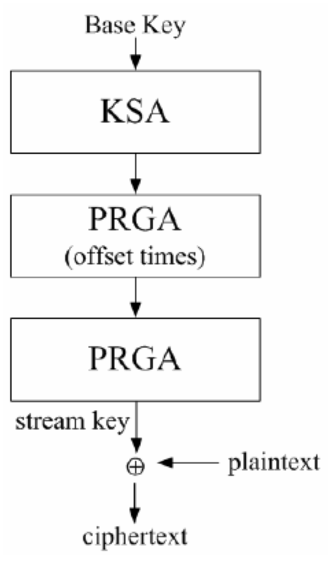
LSB algorithm : Least significant bit (LSB) coding is the simplest way to embed information in a digital audio file. By substituting the least significant bit of each sampling point with a binary message, LSB coding allows for a large amount of data to be encoded.

2.4 Video Steganography

It is a technique of hiding any kind of data into a digital video file. In this case video (combination of pictures) is used as a carrier for hiding the data.

RC4 Encryption Algorithm: **RC4** (also known as Rivest Cipher 4) is a stream cipher and variable-length key algorithm. This algorithm encrypts one byte at a time (or larger units at a time).

Flowchart For the algorithm Procedure:

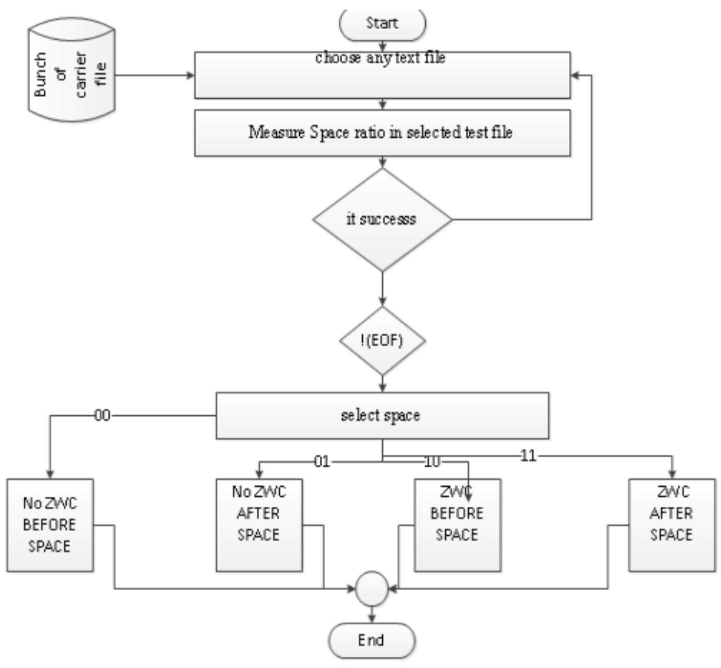


2.5 Text Steganography

Text is one of the oldest choices of media for performing steganography. The abundance of textual information on the Internet has encouraged steganography to use this media as a carrier for hidden messages.

ZWC(Zero Width Character): ZWC it's a Unicode character (U+200B), that does not occupy any space or file formatting. By adding ZWC before and after space letters can hide data.

Flowchart For the algorithm Procedure:



2.6 Image Steganography

Hiding the data by taking the cover object as an image is referred to as image steganography. In image steganography pixel intensities are used to hide the data. In digital steganography, images are widely used to cover sources because there are a number of bits present in the digital representation of an image.

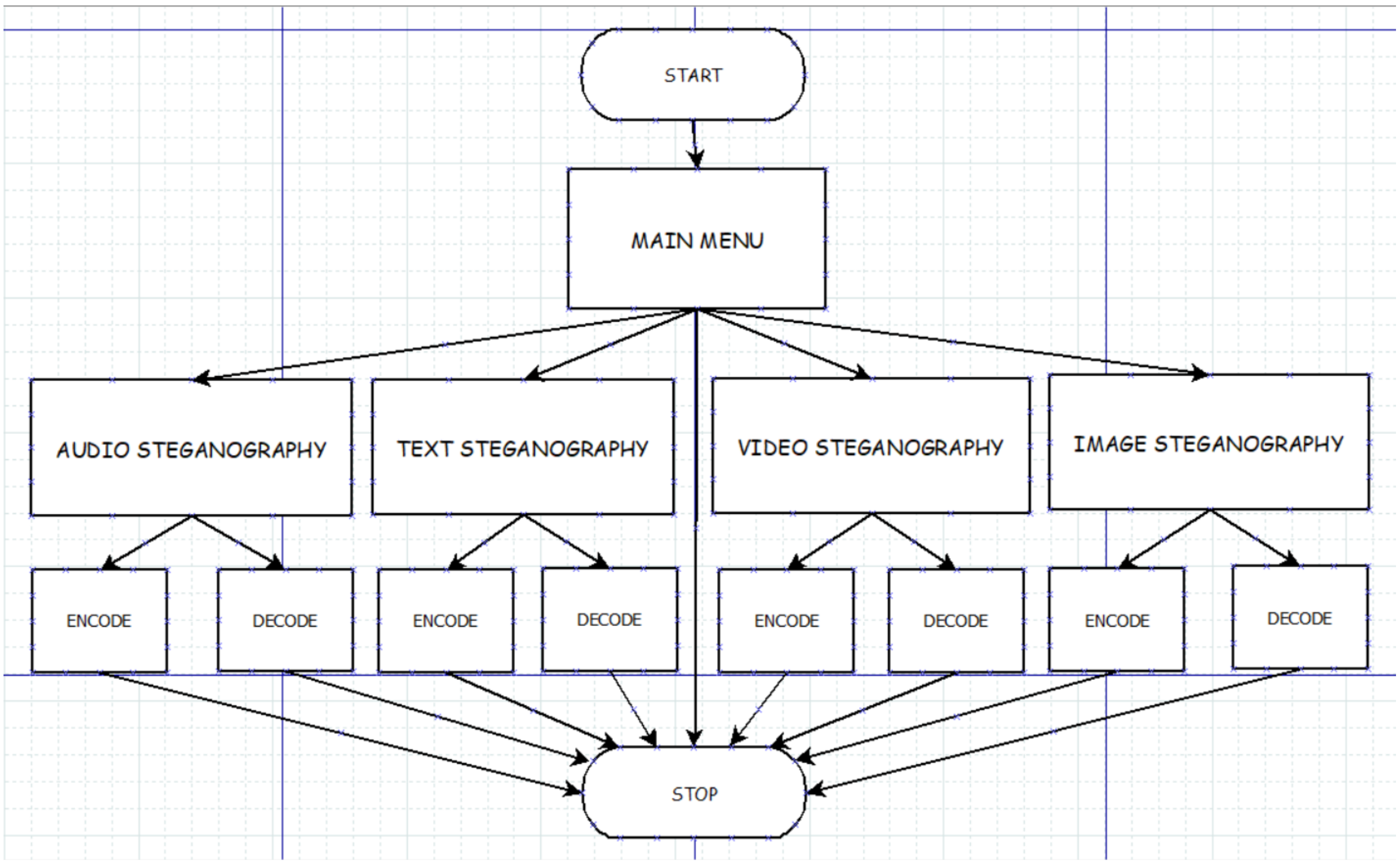
CHAPTER 3
METHODOLOGY

3.1 Introduction

This chapter contains section 3.2 depicting the overall flowchart of the program which is the step by step of overall formation of the steganographic file. Section 3.3 explained the program methodology applied in this chapter. Section 3.4 contains hardware and software requirements for **"Steganography Tools Visualizer"**

3.2 Overview

This chapter contains the flow chart of the whole application. This chapter will explain every step involved in the formation of the application.



Overall Flow Chart of The Program.

3.3 Methodology

3.31 Splitting Of the program file

We have splitted the program into two parts:

- Steganography.py (The main file)
- libs (A folder that contains the individual files as library)
    - audio_steganography.py
    - video_steganography.py
    - text_steganography.py
    - image_steganography.py

3.32 Creating "audio_steganography.py"

<u>This file contains a function called:</u>

- Audio_steganography(file,n):

<u>with parameters :</u> "file" - which takes the file name as argument.

"n" - which takes a integer i.e.: 0 or 1 as argument, which acts as a flag, means 0 for encode and 1 for decode.

<u>The Sub-functions:</u>

- Encode():
- Decode():

<u>About Sub-functions:</u>

<u>Encode()</u> : In this function we will be using Cover Audio as a Cover file to encode the given text. Wave module is used to read the audio file. Firstly we convert our secret message to its binary equivalent and added delimiter '*****' to the end of the message. For encoding we have modified the LSB Algorithm, for that we take each frame byte of the converting it to 8 bit format then check for the 4th LSB and see if it matches with the secret message bit. If yes, change the 2nd LSB to 0 using the logical AND operator between each frame byte and 253(11111101). Else we change the 2nd LSB to 1 using logical AND operation with 253 and then logical OR to change it to 1 and now add secret message bit in LSB for achieving that use logical AND operation between each frame byte of carrier audio and a binary number of

254 (11111110). Then logical OR operation between modified carrier byte and the next bit (0 or 1) from the secret message which resets the LSB of the carrier byte.

Decode() : In this function we start the extraction process by reading each frame and converting it to byte array. After that we check the 2nd LSB if it is 0 or 1. If the bit is 1 we store the LSB of the frame byte else we store the 4th LSB, we keep this process until we reach the delimiter and then we break from the loop, then convert the message into characters and print it.

3.33 Creating “video_steganography.py”

This file contains a function called:

- Video_steganography(file,n):

with parameters : “file” - which takes the file name as argument.

“n” - which takes a integer i.e.: 0 or 1 as argument, which acts as a flag, means 0 for encode and 1 for decode.

The Sub-functions:

- Encode():
- Decode():

About Sub-functions:

Encode() : In video steganography we have used a combination of cryptography and Steganography. We encode the message through two parts:

- We convert plaintext to cipher text for doing so we have used RC4 Encryption Algorithm. RC4 is a stream cipher and variable-length key algorithm. This algorithm encrypts one byte at a time. It has two major parts for encryption and decryption:-

KSA(Key-Scheduling Algorithm) : A list S of length 256 is made and the entries of S are set equal to the values from 0 to 255 in ascending order. We ask the user for a key and convert it to its equivalent ascii code. S[] is a permutation of 0,1,2....255, now a variable j is assigned as j=(j+S[i]+key[i%key_length) mod 256 and swap S(i) with S(j) and accordingly we get new permutation for the whole keystream according to the key.

PRGA(Pseudo random generation Algorithm (Stream Generation)) : Now we take input length of plaintext and initiate a loop to generate a keystream byte of equal length. For this we initiate i=0, j=0 now increment i by 1 and mod with 256. Now we add S[i] to j and mod it with 256 ,again swap the values. At the last step, take store keystream bytes which match as S[(S[i]+S[j]) mod 256] to finally get a key stream of length same as plaintext.

Now we xor the plaintext with keystream to get the final cipher.

- Now for the Steganography part we will be using Modified LSB Algorithm where we overwrite the LSB bits of the selected frame (given by the user) from the cover video, with the bit of text message character. At the end of the text message we add a delimiter to the message string as an endpoint which comes useful in the decoding function. We encode data in order of Red, then Green and then Blue pixel for the entire message of the selected frame.

Decode() : In decode part In the decode part, we take the encoded frame from the stego video, in the frame each pixels last LSB is stored until we get to the delimiter as we reach there we split them by 8 bits and convert them to characters data type now we go to the decryption process where we do the same as encode, make Keystream with help of secret key and using KSA and PRGA and finally xoring with the obtained data from the frame with keystream to get the final decoded secret message.

3.34 Creating "text_steganography.py"

This file contains a function called:

- Text_steganography(file,n):

with parameters : "file" - which takes the file name as argument.

"n" - which takes a integer i.e.: 0 or 1 as argument, which acts as a flag, means 0 for encode and 1 for decode.

The Sub-functions:

- Encode():
- Decode():

About Sub-functions:

Encode() :

ZWCs : In Unicode, there are specific zero-width characters (ZWC). We used four ZWCs for hiding the SM through the CT.

The embedding algorithm contains following stages:-

Secret Message (SM) : This can be secret or confidential information.

Cover Text (CT) : This is an innocent text that can be any type of meaningful text.

For every character of the secret message :-

- We get its ascii value and it is incremented or decremented based on if ascii value between 32 and 64 , it is incremented by 48(ascii value for 0) else it is decremented by 48.
- Then xor the the obtained value with 170(binary equivalent-10101010)
- Convert the obtained number from the first two steps to its binary equivalent then add "0011" if it earlier belonged to ascii value between 32 and 64 else add "0110" making it 12 bit for each character.

With the final binary equivalent we also use 111111111111 as a delimiter to find the end of a message.

Now from 12 bits representing each character every 2 bits is replaced with equivalent ZWCs according to the table given below. Each character is hidden after a word in the cover text.

2 bit classification	Hexcode
00	0x200
01	0x202C
11	0x202D
10	0x200E

Decode() : After receiving a stego file , the extraction algorithm discovers the contractual 2-bit of each ZWCs , every 12 bit from end of the word in the stego file and then the binary equivalent is completely extracted and delimiter discussed above helps us in getting to the end point. Now we divide the 12 bit into two parts first 4 bit and another 8bit on which we do the xor operation with 170(binary value 10101010). Now according to the first 4 bit if it is "0110" we increment it by 48 else we decrement by 48. At last we convert the ascii value into its equivalent character to get the final hidden message from the stego file.

3.35 Creating "image_steganography.py"

This file contains a function called:

- image_steganography(file,n):

with parameters : "file" - which takes the file name as argument.

"n" - which takes a integer i.e.: 0 or 1 as argument, which acts as a flag, means 0 for encode and 1 for decode.

The Sub-functions:

- Encode():
- Decode():

About Sub-functions:

Encode(): Using Modified LSB Algorithm where we overwrite the LSB bit of actual image with the bit of text message character. At the end of the text message we push a delimiter to the message string as a checkpoint useful in the decoding function. We encode data in order of Red, then Green and then Blue pixel for the entire message.

Decode(): In the decode part, we take all the LSB bits of each pixel until we get a checkpoint/delimiter and then we split them by 8 bits and convert them to characters data type and print the string (i.e., the secret text message) without delimiter.

3.36 Creating "Steganography.py"

This is the main file of our program. This executes a terminal application according to the format given in the help menu and connects all the "/lib" files to one main file, making the application faster.

3.4 Software hardware requirements

Ram : 4 Gb
Operating System : Linux OS (Only)
Programming Language : Python
Software :

- VMware (If the base system is not Linux)
- PyCharm (IDE for Python Programming Language)
- Git,Github

CHAPTER 4

DESIGN AND IMPLEMENTATION

4.1 Introduction

This chapter contains section 4.2 depicting the cover file format of the steganographic file. Section 4.3 explains how to implement the program **"Steganography Tools Visualizer"**.

4.2 The cover file format

The cover file formats are :

● Audio cover file	:	.wav (only)
● Video cover file	:	.mp4 / .mkv
● Text cover file	:	.txt
● Image cover file	:	.jpg / .jpeg / png

4.3 How To Implement

Pre-requirements :

- Clone the project :

  To clone the project use the command :

  - ➔ git clone https://github.com/Sanjipan/Steganography
- Install Python Libraries :

  We have to install the following python libraries with the commands :

  - ➔ pip install argparse
  - ➔ pip install Wave
  - ➔ pip install opencv-python
  - ➔ pip install numpy
  - ➔ pip install Pillow
  - ➔ pip install pytest-shutil
  - ➔ pip install subprocess.run
  - ➔ pip install stegano
- To use the code :

  Navigate to the Directory :

➔ cd Steganography

<u>Format to use the application</u>:

➔ sudo python3 ./Steganography.py <**File Type**> <**Encode/Decode**> <**File Location**>

Implementing the code :

<u>For Audio Cover File</u> :

# Encoding :

➔ sudo python3 ./Steganography.py -a -e <location of file>

➔ sudo python3 ./Steganography.py --audio --encode <location of file>

# Decoding :

➔ sudo python3 ./Steganography.py -a -d <location of file>

➔ sudo python3 ./Steganography.py --audio --decode <location of file>

<u>For Video Cover File</u> :

# Encoding :

➔ sudo python3 ./Steganography.py -v -e <location of file>

➔ sudo python3 ./Steganography.py --video --encode <location of file>

# Decoding :

➔ sudo python3 ./Steganography.py -v -d <location of file>

➔ sudo python3 ./Steganography.py --video --decode <location of file>

<u>For Image Cover File</u> :

# Encoding :

➔ sudo python3 ./Steganography.py -i -e <location of file>

➔ sudo python3 ./Steganography.py --image --encode <location of file>

# Decoding :

➔ sudo python3 ./Steganography.py -i -d <location of file>

➔ sudo python3 ./Steganography.py --image --decode <location of file>

For Text Cover File :

# Encoding :

➔ sudo python3 ./Steganography.py -t -e <location of file>

➔ sudo python3 ./Steganography.py --text --encode <location of file>

# Decoding :

➔ sudo python3 ./Steganography.py -t -d <location of file>

➔ sudo python3 ./Steganography.py --text --decode <location of file>

For Help :

➔ sudo python3 ./Steganography.py -h

➔ sudo python3 ./Steganography.py --help

CHAPTER 5

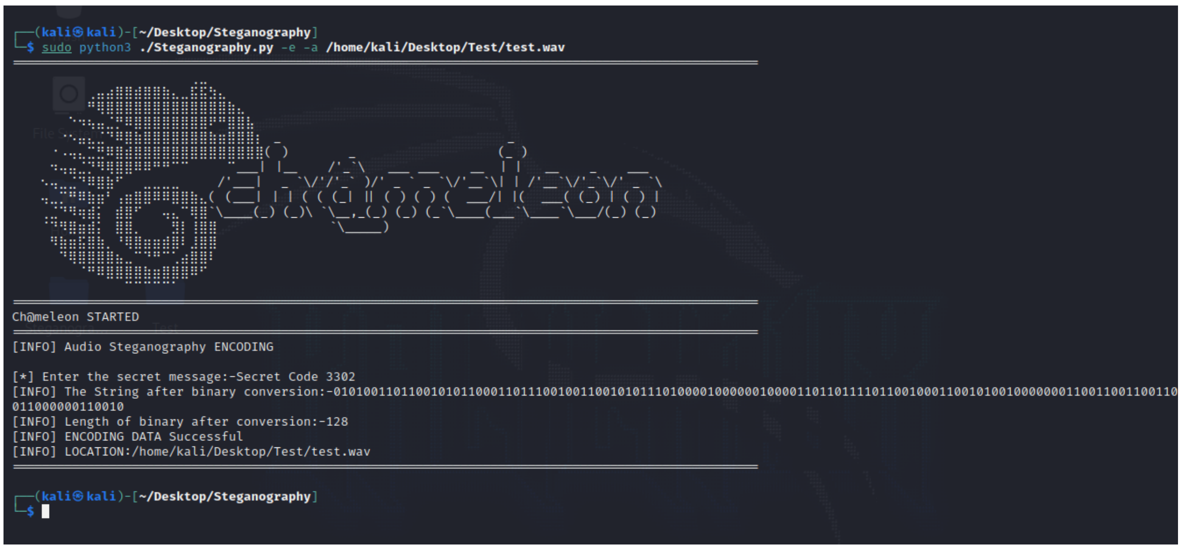
RESULTS AND DISCUSSION

5.1 Introduction

This chapter contains section 5.2 depicting the Result Analysis of Audio Steganography, section 5.3 depicting the Result Analysis of Video Steganography, section 5.4 depicting the Result Analysis of Text Steganography and section 5.5 depicting the Result Analysis of Image Steganography.

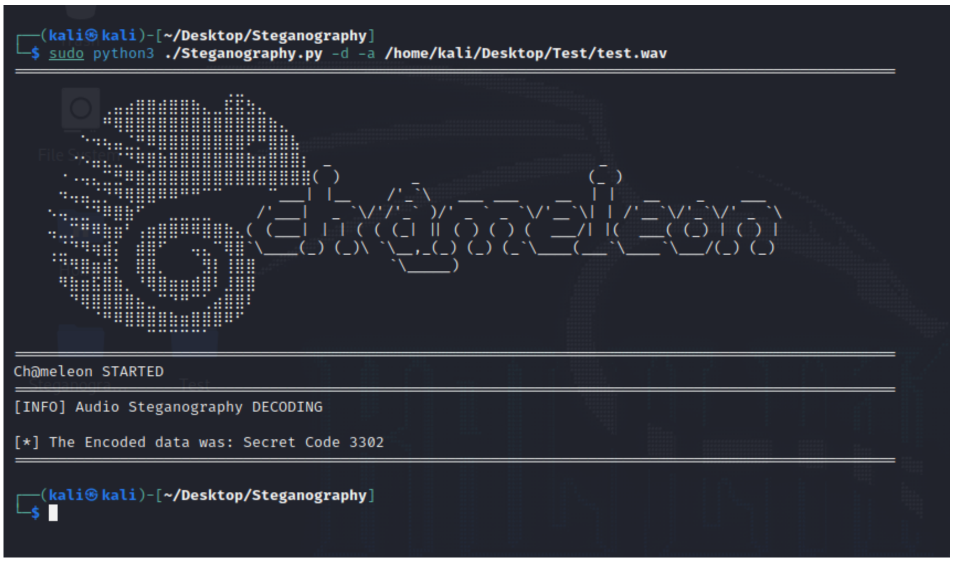
5.2 Result Analysis - Audio Steganography

Audio cover file encoding :



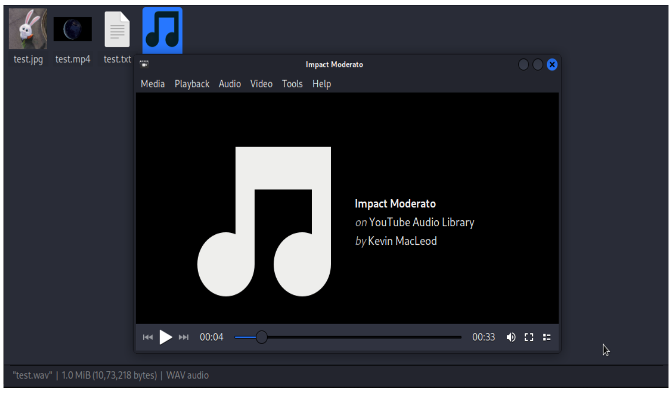
```
┌──(kali㉿kali)-[~/Desktop/Steganography]
└─$ sudo python3 ./Steganography.py -e -a /home/kali/Desktop/Test/test.wav

Ch@meleon STARTED

[INFO] Audio Steganography ENCODING

[*] Enter the secret message:-Secret Code 3302
[INFO] The String after binary conversion:-01010011011001010110001101110010011001010111010000100000010000110110111101100100011001010010000000110011001100110
011000000110010
[INFO] Length of binary after conversion:-128
[INFO] ENCODING DATA Successful
[INFO] LOCATION:/home/kali/Desktop/Test/test.wav

┌──(kali㉿kali)-[~/Desktop/Steganography]
└─$ 
```

Audio cover file decoding :

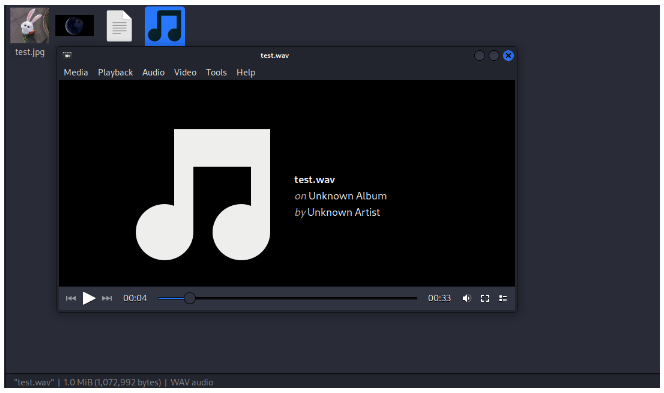


```
┌──(kali㉿kali)-[~/Desktop/Steganography]
└─$ sudo python3 ./Steganography.py -d -a /home/kali/Desktop/Test/test.wav

Ch@meleon STARTED

[INFO] Audio Steganography DECODING

[*] The Encoded data was: Secret Code 3302

┌──(kali㉿kali)-[~/Desktop/Steganography]
└─$ 
```

<u>test.wav cover file before</u> :



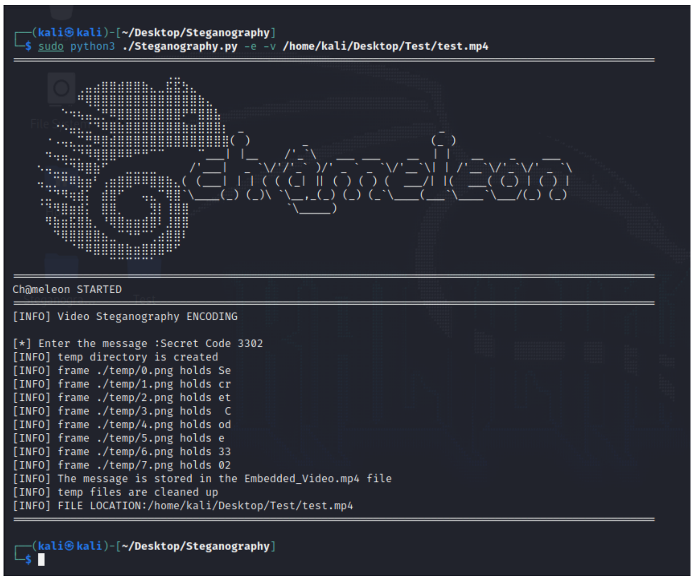
<u>test.wav cover file after</u> :



test.jpg
test.wav
Media Playback Audio Video Tools Help
test.wav
on Unknown Album
by Unknown Artist
00:04
00:33
"test.wav" | 1.0 MiB (1,072,992 bytes) | WAV audio

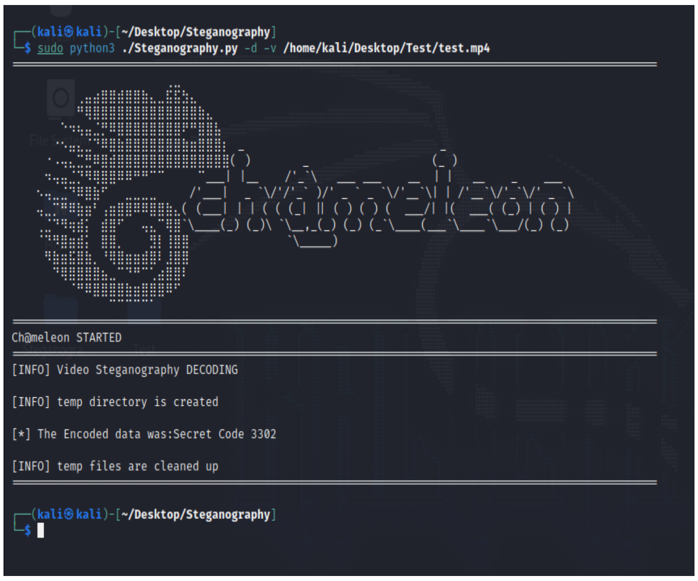
5.3 Result Analysis - Video Steganography

<u>Video cover file encoding</u> :



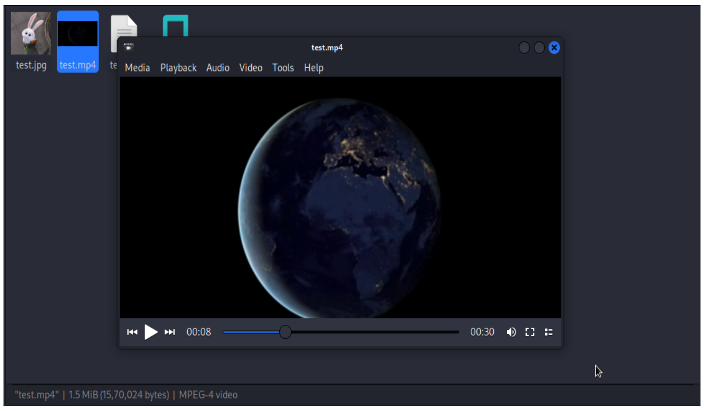
```
┌──(kali㉿kali)-[~/Desktop/Steganography]
└─$ sudo python3 ./Steganography.py -e -v /home/kali/Desktop/Test/test.mp4

Ch@meleon STARTED

[INFO] Video Steganography ENCODING

[*] Enter the message :Secret Code 3302
[INFO] temp directory is created
[INFO] frame ./temp/0.png holds Se
[INFO] frame ./temp/1.png holds cr
[INFO] frame ./temp/2.png holds et
[INFO] frame ./temp/3.png holds  C
[INFO] frame ./temp/4.png holds od
[INFO] frame ./temp/5.png holds e
[INFO] frame ./temp/6.png holds 33
[INFO] frame ./temp/7.png holds 02
[INFO] The message is stored in the Embedded_Video.mp4 file
[INFO] temp files are cleaned up
[INFO] FILE LOCATION:/home/kali/Desktop/Test/test.mp4

┌──(kali㉿kali)-[~/Desktop/Steganography]
└─$ 
```

<u>Video cover file decoding</u> :



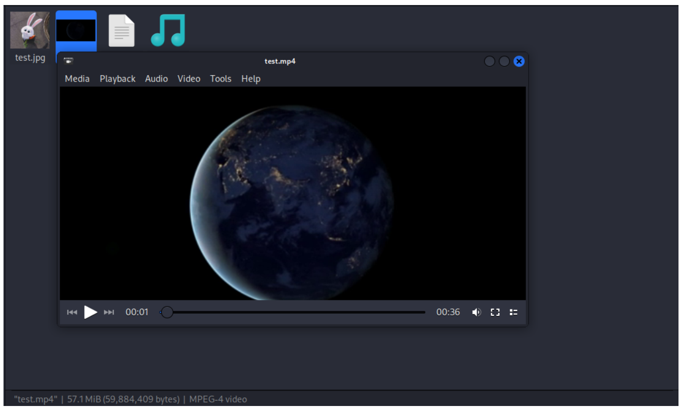
```
┌──(kali㉿kali)-[~/Desktop/Steganography]
└─$ sudo python3 ./Steganography.py -d -v /home/kali/Desktop/Test/test.mp4

Ch@meleon STARTED

[INFO] Video Steganography DECODING

[INFO] temp directory is created

[*] The Encoded data was:Secret Code 3302

[INFO] temp files are cleaned up

┌──(kali㉿kali)-[~/Desktop/Steganography]
└─$ 
```

<u>test.mp4 cover file before</u> :



test.jpg
test.mp4
test.mp4
Media Playback Audio Video Tools Help
00:08
00:30
"test.mp4" | 1.5 MiB (15,70,024 bytes) | MPEG-4 video

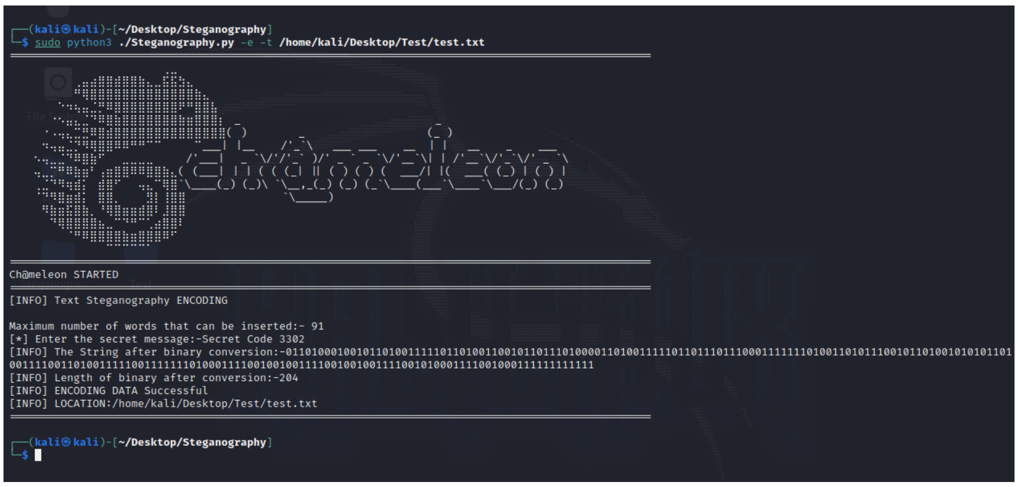
<u>test.mp4 cover file after</u> :



test.jpg
test.mp4
Media Playback Audio Video Tools Help
00:01
00:36
"test.mp4" | 57.1 MiB (59,884,409 bytes) | MPEG-4 video

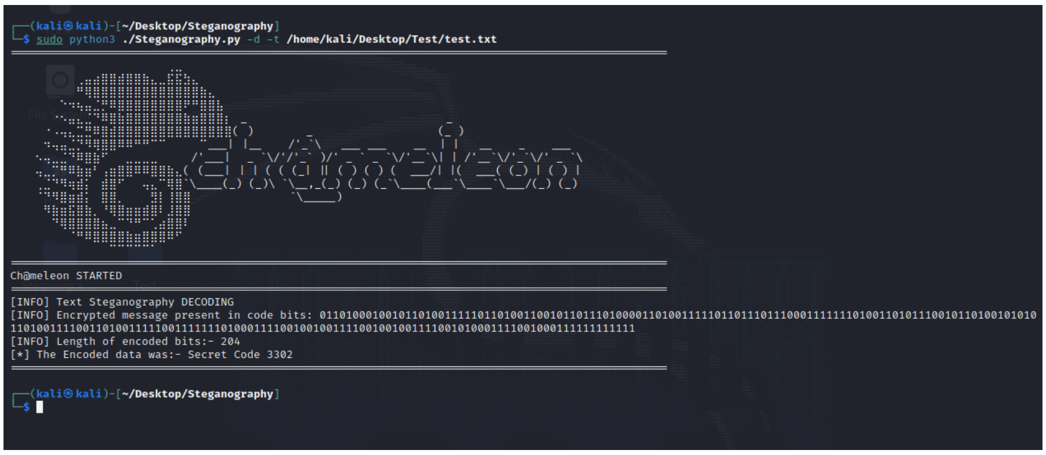
5.4 Result Analysis - Text Steganography

Text cover file encoding :



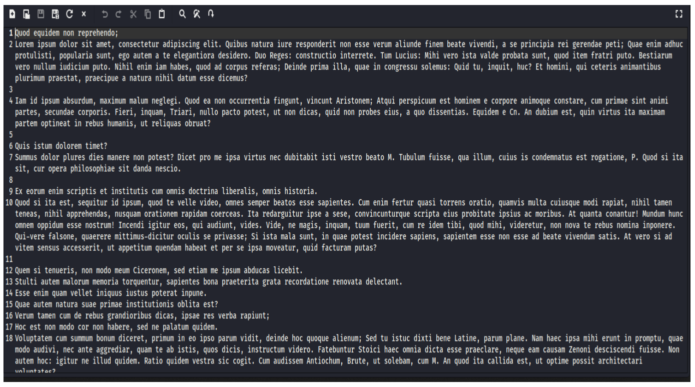
```
┌──(kali㉿kali)-[~/Desktop/Steganography]
└─$ sudo python3 ./Steganography.py -e -t /home/kali/Desktop/Test/test.txt

Ch@meleon STARTED

[INFO] Text Steganography ENCODING

Maximum number of words that can be inserted:- 91
[*] Enter the secret message:-Secret Code 3302
[INFO] The String after binary conversion:-0110100010010110100111110110100110010110111010000110100111110110111011100011111110100110101110010110100101010110100111100110100111110011111110100011110010010011110010010011110010100011110010001111111111
[INFO] Length of binary after conversion:-204
[INFO] ENCODING DATA Successful
[INFO] LOCATION:/home/kali/Desktop/Test/test.txt

┌──(kali㉿kali)-[~/Desktop/Steganography]
└─$ 
```

Text cover file decoding :



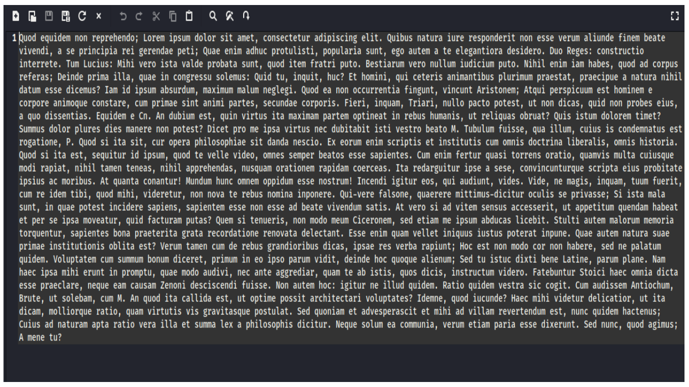
```
┌──(kali㉿kali)-[~/Desktop/Steganography]
└─$ sudo python3 ./Steganography.py -d -t /home/kali/Desktop/Test/test.txt

Ch@meleon STARTED

[INFO] Text Steganography DECODING
[INFO] Encrypted message present in code bits: 01101000100101101001111101101001100101101110100001101001111101101110111000111111101001101011100101101001010101101001111001101001111100111111101000111100100100111100100100111100101000111100100011111111111
[INFO] Length of encoded bits:- 204
[*] The Encoded data was:- Secret Code 3302

┌──(kali㉿kali)-[~/Desktop/Steganography]
└─$ 
```

test.txt cover file before :



```
Quod equidem non reprehendo;
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quibus natura iure responderit non esse verum aliunde finem beate vivendi, a se principia rei gerendae peti; Quae enim adhuc protulisti, popularia sunt, ego autem a te elegantiora desidero. Duo Reges: constructio interrete. Tum Lucius: Mihi vero ista valde probata sunt, quod item fratri puto. Bestiarum vero nullum iudicium puto. Nihil enim iam habes, quod ad corpus referas; Deinde prima illa, quae in congressu solemus: Quid tu, inquit, huc? Et homini, qui ceteris animantibus plurimum praestat, praecipue a natura nihil datum esse dicemus?

Iam id ipsum absurdum, maximum malum neglegi. Quod ea non occurrentia fingunt, vincunt Aristonem; Atqui perspicuum est hominem e corpore animoque constare, cum primae sint animi partes, secundae corporis. Fieri, inquam, Triari, nullo pacto potest, ut non dicas, quid non probes eius, a quo dissentias. Equidem e Cn. An dubium est, quin virtus ita maximam partem optineat in rebus humanis, ut reliquas obruat?

Quis istum dolorem timet?
Summus dolor plures dies manere non potest? Dicet pro me ipsa virtus nec dubitabit isti vestro beato M. Tubulum fuisse, qua illum, cuius is condemnatus est rogatione, P. Quod si ita sit, cur opera philosophiae sit danda nescio.

Ex eorum enim scriptis et institutis cum omnis doctrina liberalis, omnis historia.
Quod si ita est, sequitur id ipsum, quod te velle video, omnes semper beatos esse sapientes. Cum enim fertur quasi torrens oratio, quamvis multa cuiusque modi rapiat, nihil tamen teneas, nihil apprehendas, nusquam orationem rapidam coerceas. Ita redarguitur ipse a sese, convincunturque scripta eius probitate ipsius ac moribus. At quanta conantur! Mundum hunc omnem oppidum esse nostrum! Incendi igitur eos, qui audiunt, vides. Vide, ne magis, inquam, tuum fuerit, cum re idem tibi, quod mihi, videretur, non nova te rebus nomina inponere. Qui-vere falsone, quaerere mittimus-dicitur oculis se privasse; Si ista mala sunt, in quae potest incidere sapiens, sapientem esse non esse ad beate vivendum satis. At vero si ad vitem sensus accesserit, ut appetitum quendam habeat et per se ipsa moveatur, quid facturam putas?

Quem si tenueris, non modo meum Ciceronem, sed etiam me ipsum abducas licebit.
Stulti autem malorum memoria torquentur, sapientes bona praeterita grata recordatione renovata delectant.
Esse enim quam vellet iniquus iustus poterat inpune.
Quae autem natura suae primae institutionis oblita est?
Verum tamen cum de rebus grandioribus dicas, ipsae res verba rapiunt;
Hoc est non modo cor non habere, sed ne palatum quidem.
Voluptatem cum summum bonum diceret, primum in eo ipso parum vidit, deinde hoc quoque alienum; Sed tu istuc dixti bene Latine, parum plane. Nam haec ipsa mihi erunt in promptu, quae modo audivi, nec ante aggrediar, quam te ab istis, quos dicis, instructum videro. Fatebuntur Stoici haec omnia dicta esse praeclare, neque eam causam Zenoni desciscendi fuisse. Non autem hoc: igitur ne illud quidem. Ratio quidem vestra sic cogit. Cum audissem Antiochum, Brute, ut solebam, cum M. An quod ita callida est, ut optime possit architectari voluptates?
```

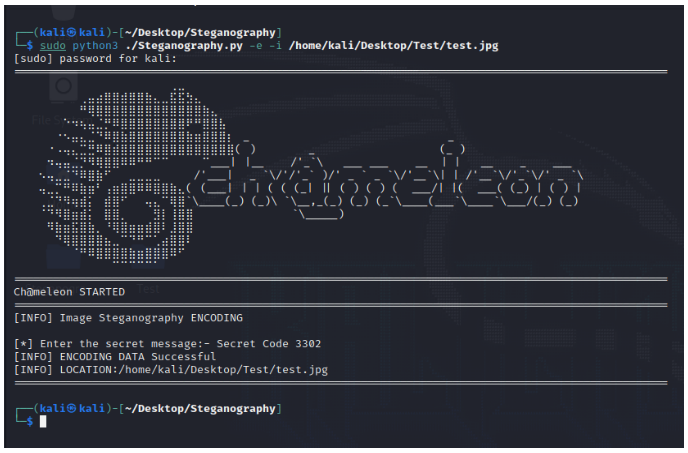
test.txt cover file after :



```
Quod equidem non reprehendo; Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quibus natura iure responderit non esse verum aliunde finem beate vivendi, a se principia rei gerendae peti; Quae enim adhuc protulisti, popularia sunt, ego autem a te elegantiora desidero. Duo Reges: constructio interrete. Tum Lucius: Mihi vero ista valde probata sunt, quod item fratri puto. Bestiarum vero nullum iudicium puto. Nihil enim iam habes, quod ad corpus referas; Deinde prima illa, quae in congressu solemus: Quid tu, inquit, huc? Et homini, qui ceteris animantibus plurimum praestat, praecipue a natura nihil datum esse dicemus? Iam id ipsum absurdum, maximum malum neglegi. Quod ea non occurrentia fingunt, vincunt Aristonem; Atqui perspicuum est hominem e corpore animoque constare, cum primae sint animi partes, secundae corporis. Fieri, inquam, Triari, nullo pacto potest, ut non dicas, quid non probes eius, a quo dissentias. Equidem e Cn. An dubium est, quin virtus ita maximam partem optineat in rebus humanis, ut reliquas obruat? Quis istum dolorem timet? Summus dolor plures dies manere non potest? Dicet pro me ipsa virtus nec dubitabit isti vestro beato M. Tubulum fuisse, qua illum, cuius is condemnatus est rogatione, P. Quod si ita sit, cur opera philosophiae sit danda nescio. Ex eorum enim scriptis et institutis cum omnis doctrina liberalis, omnis historia. Quod si ita est, sequitur id ipsum, quod te velle video, omnes semper beatos esse sapientes. Cum enim fertur quasi torrens oratio, quamvis multa cuiusque modi rapiat, nihil tamen teneas, nihil apprehendas, nusquam orationem rapidam coerceas. Ita redarguitur ipse a sese, convincunturque scripta eius probitate ipsius ac moribus. At quanta conantur! Mundum hunc omnem oppidum esse nostrum! Incendi igitur eos, qui audiunt, vides. Vide, ne magis, inquam, tuum fuerit, cum re idem tibi, quod mihi, videretur, non nova te rebus nomina inponere. Qui-vere falsone, quaerere mittimus-dicitur oculis se privasse; Si ista mala sunt, in quae potest incidere sapiens, sapientem esse non esse ad beate vivendum satis. At vero si ad vitem sensus accesserit, ut appetitum quendam habeat et per se ipsa moveatur, quid facturam putas? Quem si tenueris, non modo meum Ciceronem, sed etiam me ipsum abducas licebit. Stulti autem malorum memoria torquentur, sapientes bona praeterita grata recordatione renovata delectant. Esse enim quam vellet iniquus iustus poterat inpune. Quae autem natura suae primae institutionis oblita est? Verum tamen cum de rebus grandioribus dicas, ipsae res verba rapiunt; Hoc est non modo cor non habere, sed ne palatum quidem. Voluptatem cum summum bonum diceret, primum in eo ipso parum vidit, deinde hoc quoque alienum; Sed tu istuc dixti bene Latine, parum plane. Nam haec ipsa mihi erunt in promptu, quae modo audivi, nec ante aggrediar, quam te ab istis, quos dicis, instructum videro. Fatebuntur Stoici haec omnia dicta esse praeclare, neque eam causam Zenoni desciscendi fuisse. Non autem hoc: igitur ne illud quidem. Ratio quidem vestra sic cogit. Cum audissem Antiochum, Brute, ut solebam, cum M. An quod ita callida est, ut optime possit architectari voluptates? Idemne, quod iucunde? Haec mihi videtur delicatior, ut ita dicam, molliorque ratio, quam virtutis vis gravitasque postulat. Sed quoniam et advesperascit et mihi ad villam revertendum est, nunc quidem hactenus; Cuius ad naturam apta ratio vera illa et summa lex a philosophis dicitur. Neque solum ea communia, verum etiam paria esse dixerunt. Sed nunc, quod agimus; A mene tu?
```

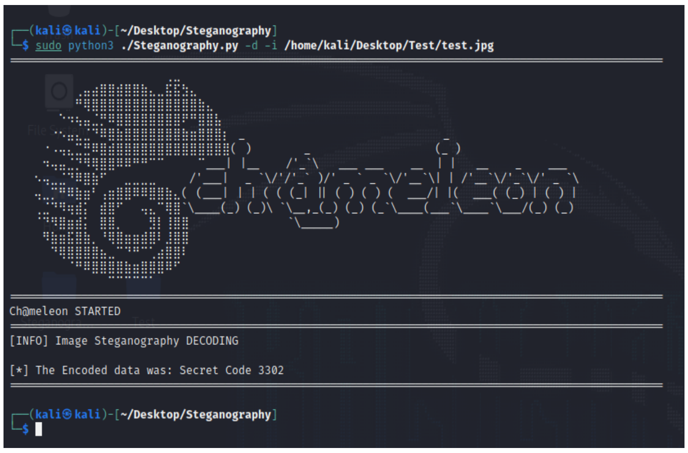
5.5 Result Analysis - Image Steganography

Image cover file encoding :



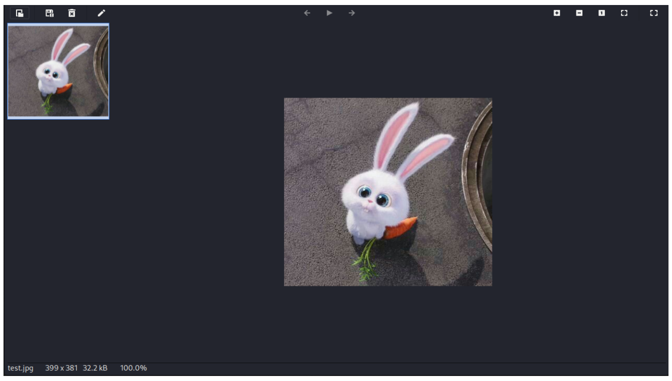
```
┌──(kali㉿kali)-[~/Desktop/Steganography]
└─$ sudo python3 ./Steganography.py -e -i /home/kali/Desktop/Test/test.jpg
[sudo] password for kali:

Ch@meleon STARTED

[INFO] Image Steganography ENCODING

[*] Enter the secret message:- Secret Code 3302
[INFO] ENCODING DATA Successful
[INFO] LOCATION:/home/kali/Desktop/Test/test.jpg

┌──(kali㉿kali)-[~/Desktop/Steganography]
└─$ █
```

Image cover file decoding :

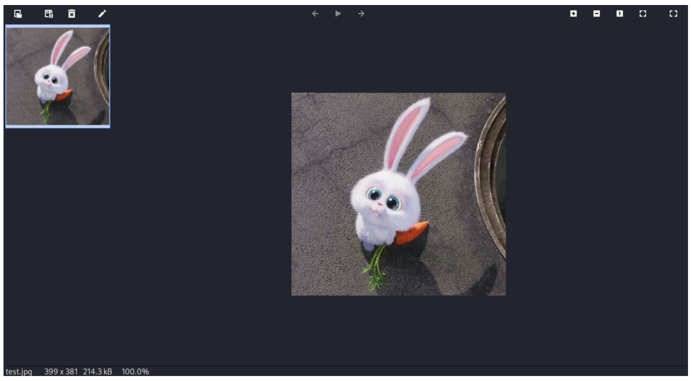


```
┌──(kali㉿kali)-[~/Desktop/Steganography]
└─$ sudo python3 ./Steganography.py -d -i /home/kali/Desktop/Test/test.jpg

Ch@meleon STARTED

[INFO] Image Steganography DECODING

[*] The Encoded data was: Secret Code 3302

┌──(kali㉿kali)-[~/Desktop/Steganography]
└─$ █
```

test.jpg cover file before :



test.jpg cover file after :



CONCLUSION

This report for the project **“Steganography Tools Visualizer”** has a systematic review on the methodology and results of the project **“Steganography Tools Visualizer”** along with the description and history of steganography. This paper also includes information on the four types of steganography i.e: audio, video, text, image and also how to implement the project.

The advantage of steganography over cryptography 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 and its contents.

This project is a success over making a fusion of old tradition of steganography with new technologies to create a safe and secure tool to hide sensitive information under an innocent looking cover file to increase the security of confidential messages across a network. The scope of this project is to include all significant current security efforts and the old tradition of handling confidential messages to create a tool that may allow users to secure their confidential messages.

REFERENCES

- *Steganography*, 2020, [online] Available: https://en.wikipedia.org/wiki/Steganography.
- H. Shi, X.-Y. Zhang, S. Wang, G. Fu and J. Tang, "Synchronized detection and recovery of steganographic messages with adversarial learning", *Proc. Int. Conf. Comput. Sci*, pp. 31-43, 2019.
- N. F. Hordri, S. S. Yuhaniz and S. M. Shamsuddin, "Deep learning and its applications: A review", *Proc. Conf. Postgraduate Annu. Res. Informat. Seminar*, pp. 1-6, 2016.
- N. F. Johnson and S. Jajodia, "Exploring steganography: Seeing the unseen", *Computer*, vol. 31, no. 2, pp. 26-34, Feb. 1998.
- S. Gupta, G. Gujral and N. Aggarwal, "Enhanced least significant bit algorithm for image steganography", *Int. J. Comput. Eng. Manage.*, vol. 15, no. 4, pp. 40-42, 2012.
- R. J. Mstafa, K. M. Elleithy and E. Abdelfattah, "A robust and secure video steganography method in DWT-DCT domains based on multiple object tracking and ECC", *IEEE Access*, vol. 5, pp. 5354-5365, 2017.

Appendix

a.1 Full Coding

File Steganography.py :

```
import sys
import argparse

from libs import audio_steganography
from libs import image_steganography
from libs import text_steganography
from libs import video_steganography

use_text_encode = "python3 ./Steganography.py -t -e <location of file>"
use_text_decode = "python3 ./Steganography.py -t -d <location of file>"

use_audio_encode = "python3 ./Steganography.py -a -e <location of file>"
use_audio_decode = "python3 ./Steganography.py -a -d <location of file>"

use_video_encode = "python3 ./Steganography.py -v -e <location of file>"
use_video_decode = "python3 ./Steganography.py -v -d <location of file>"

use_image_encode = "python3 ./Steganography.py -i -e <location of file>"
use_image_decode = "python3 ./Steganography.py -i -d <location of file>"

use_help = "python3 ./Steganography.py -h -----> help"



def symbols():
    print("=" * 100)
    print("                                  ⢀⣀                ")
    print("                  ⢀⣤⣴⣿⣿⣾⣿⣿⣷⣄⣀⣟⣟⢻⣄          ")
    print("                  ⠛⡿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣷⣄      ")
    print("               ⠑⠪⠳⣤⣠⡴⠟⠟⣿⣿⣿⣿⣿⣿⣿⣿⡟⠛⣿⣿⣷    ")
    print("             ⠈⠑⣤⣤⣀⡠⠚⠛⣿⣷⣿⣿⣿⣿⣿⣿⣿⣷⣴⣿⣿⣿⡇   _                    _")
    print("        ⠂ ⠠⠒⠢⠤⠤⠭⠭⠟⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿( )      _              (_)")
    print("       ⠒⠢⠤⠤⠔⠚⠛⠻⣿⣿⣿⠿⠿⠛⠉⠉        ⠉___|  |__   /'_`\  ___ ___   __ | |  __   _   ___")
    print("     ⠑⠢⠤⠤⠔⠙⠿⣿⣷⠋    ⠤⠤⠤⠤⠤       /'___|    _`\/'/'_` )/' _ ` _ `\/'__`\| | /'__`\/'_`\/' _ `\ ")
    print("     ⠒⠤⠔⠉⠛⠿⣷⣶⠃ ⠰⣶⣿⣿⠿⠿⣿⣿⣷⣄( (___|  | | ( ( (_| || ( ) ( ) (  ___/| |(  ___( (_) | ( ) |")
    print("      ⠠⠔⠉⠙⠻⣴⣿⡇  ⣼⣿⠋    ⠒⠤⠉⠻⣿`\____(_) (_)\ `\__,_(_) (_) (_`\____(___`\____`\___/(_) (_)")
    print("      ⠁⠙⠻⣿⣶⣼⡇  ⣿⣿⡀     ⣿⡇ ⡇⣿⣿                  `\_____)")
    print("       ⠻⣷⣶⣯⣿⣷⡀ ⠘⠻⣿⣶⣶⣼⣿⠃ ⣠⣿⣿            ")
    print("         ⠙⠻⣿⣿⣿⣿⣿⣦⣤⠉⠙⠛⠉⠂⣠⣾⣿⡟            ")
    print("              ⠈⠛⠿⣿⣿⣿⣿⣿⣷⣶⣿⣿⣿⣿⠿⠋            ")
    print("                         ⠉⠉⠉⠉⠉⠉⠁            ")
    print("=" * 100)
    print("Ch@meleon STARTED")
    print("=" * 100)



def main():
    parser = argparse.ArgumentParser()
    parser.add_argument('-a', '--audio', action='store_true', help="For audio file")
    parser.add_argument('-t', '--text', action='store_true', help="For text file")
    parser.add_argument('-v', '--video', action='store_true', help="For video file")
    parser.add_argument('-i', '--image', action='store_true', help="For image file")
    parser.add_argument('-e', '--encode', action='store_true', help="For encoding")
    parser.add_argument('-d', '--decode', action='store_true', help="For decoding")
    parser.add_argument('filename')
    args = parser.parse_args()
    c = 0
    t = True
    if args.filename:
        symbols()
        if args.audio and args.encode and t:
            audio_steganography.Audio_steganography(args.filename, 0)
            t = False
        elif args.audio and args.decode and t:
            audio_steganography.Audio_steganography(args.filename, 1)
            t = False
        elif args.text and args.encode and t:
            text_steganography.Text_steganography(args.filename, 0)
            t = False
        elif args.text and args.decode and t:
            text_steganography.Text_steganography(args.filename, 1)
            t = False
        elif args.video and args.encode and t:
            video_steganography.Video_Steganography(args.filename, 0)
            t = False
        elif args.video and args.decode and t:
            video_steganography.Video_Steganography(args.filename, 1)
            t = False
        elif args.image and args.encode and t:
            image_steganography.Image_steganography(args.filename, 0)
            t = False
        elif args.image and args.decode and t:
            image_steganography.Image_steganography(args.filename, 1)
            t = False
        else:
            c = 1
    else:
        symbols()
        parser.print_help()
        sys.exit(0)
    if c == 1:
        symbols()
        parser.print_help()
        sys.exit()


if __name__ == "__main__":
    main()
```

File audio_steganography.py :

```
import os
import wave


def Audio_steganography(file, n):
  def Encode():
    print("[INFO] Audio Steganography ENCODING")
    print("")
    song = wave.open(file, mode='rb')
    nframes = song.getnframes()
    frames = song.readframes(nframes)
    frame_list = list(frames)
    frame_byte = bytearray(frame_list)

    data = input("[*] Enter the secret message:-")

    res = ''.join(format(i, '08b') for i in bytearray(data, encoding='utf-8'))
    print("[INFO] The String after binary conversion:-{}".format(res))
    length = len(res)
    print("[INFO] Length of binary after conversion:-{}".format(length))

    data = data + '*^*^*'

    results = []
    for j in data:
      bits = bin(ord(j))[2:].zfill(8)
      results.extend([int(b) for b in bits])

    k = 0
    for i in range(0, len(results), 1):
      res = bin(frame_byte[k])[2:].zfill(8)
      if res[len(res) - 4] == results[i]:
        frame_byte[k] = (frame_byte[k] & 253)
      else:
        frame_byte[k] = (frame_byte[k] & 253) | 2
        frame_byte[k] = (frame_byte[k] & 254) | results[i]
      k = k + 1
    frame_modified = bytes(frame_byte)
    os.remove(file)
    with wave.open(file, 'wb') as fd:
        fd.setparams(song.getparams())
        fd.writeframes(frame_modified)
    print("[INFO] ENCODING DATA Successful")
    print("[INFO] LOCATION:{}".format(file))
    song.close()
    print("=" * 100)

def Decode():
    print("[INFO] Audio Steganography DECODING")
    print("")
    song = wave.open(file, mode='rb')
    nframes = song.getnframes()
    frames = song.readframes(nframes)
    frame_list = list(frames)
    frame_bytes = bytearray(frame_list)

    extracted = ""
    p = 0
    for i in range(len(frame_bytes)):
        if p == 1:
            break
        res = bin(frame_bytes[i])[2:].zfill(8)
        if res[len(res) - 2] == 0:
            extracted += res[len(res) - 4]
        else:
            extracted += res[len(res) - 1]

        all_bytes = [extracted[i: i + 8] for i in range(0, len(extracted), 8)]
        decode_data = ""
        for byte in all_bytes:
            decode_data += chr(int(byte, 2))
            if decode_data[-5:] == '*^*^*':
                print("[*] The Encoded data was:", decode_data[:-5])
                p = 1
                break
        print("=" * 100)

    if n == 0:
        Encode()
    else:
        Decode()
```

File image_steganography.py :

```
import os
import cv2
import numpy as np
from PIL import Image



def Image_steganography(file, n):
    # it converts data in binary format

    def data2binary(data):
        p = ''
        if type(data) == str:
            p = p.join([format(ord(i), '08b') for i in data])
        elif type(data) == bytes or type(data) == np.ndarray:
            p = [format(i, '08b') for i in data]
        return p

    # hide data in given img

    def hide_data(img, data):
        data += "$$"  # '$$'--> secrete key
        d_index = 0
        b_data = data2binary(data)
        len_data = len(b_data)
    # iterate pixels from image and update pixel values

    for value in img:
        for pix in value:
            r, g, b = data2binary(pix)
            if d_index < len_data:
                pix[0] = int(r[:-1] + b_data[d_index])
                d_index += 1
            if d_index < len_data:
                pix[1] = int(g[:-1] + b_data[d_index])
                d_index += 1
            if d_index < len_data:
                pix[2] = int(b[:-1] + b_data[d_index])
                d_index += 1
            if d_index >= len_data:
                break
    return img

def Encode():

    print("[INFO] Image Steganography ENCODING")
    print("")
    image = cv2.imread(file)
    img = Image.open(file, 'r')
    w, h = img.size
    data = input("[*] Enter the secret message:- ")
    if len(data) == 0:
        raise ValueError("[INFO] Empty data")
    enc_img = 'temp.png'
    enc_data = hide_data(image, data)
    cv2.imwrite(enc_img, enc_data)
    img1 = Image.open(enc_img, 'r')
    img1 = img1.resize((w, h), Image.Resampling.LANCZOS)
    # optimize with 65% quality
    if w != h:
        img1.save(enc_img, optimize=True, quality=65)
    else:
        img1.save(enc_img)
    img.close()
    img1.close()
    os.remove(file)
    os.rename(enc_img, file)
    print("[INFO] ENCODING DATA Successful")
    print("[INFO] LOCATION:{}".format(file))
    print("=" * 100)

# decoding

def find_data(img):
    bin_data = ""
    for value in img:
        for pix in value:
            r, g, b = data2binary(pix)
            bin_data += r[-1]
            bin_data += g[-1]
            bin_data += b[-1]

    all_bytes = [bin_data[i: i + 8] for i in range(0, len(bin_data), 8)]

    readable_data = ""
    for i in all_bytes:
        readable_data += chr(int(i, 2))
        if readable_data[-2:] == "$$":
            break
    return readable_data[:-2]

def Decode():

    print("[INFO] Image Steganography DECODING")
    print("")
    image = cv2.imread(file)
    img = Image.open(file, 'r')
```

```
        msg = find_data(image)
        img.close()
        print("[*] The Encoded data was: {}".format(msg))
        print("=" * 100)

    if n == 0:
        Encode()
    else:
        Decode()
```

File text_steganography.py :

```
import os



def Text_steganography(file, n):
    def EncodeTheText(text):
        i = 0
        add = ''
        while i < len(text):
            t = ord(text[i])
            if 32 <= t <= 64:
                t1 = t + 48
                t2 = t1 ^ 170
                res = bin(t2)[2:].zfill(8)
                add = add + "0011" + res
            else:
                t1 = t - 48
                t2 = t1 ^ 170
                res = bin(t2)[2:].zfill(8)
                add = add + "0110" + res
            i = i + 1
        res1 = add + "111111111111"
        print("[INFO] The String after binary conversion:-{}".format(res1))
        print("[INFO] Length of binary after conversion:-{}".format(len(res1)))
        ZWC = {"00": u'\u200C', "01": u'\u202C', "11": u'\u202D', "10": u'\u200E'}
```

```
file1 = open(file, 'r+')
f = file.split("/")
f[len(f)-1] = "/temp.txt"
aa = file.split("/")
a = ''
for i in range(1, len(aa) - 1):
    a = a + "/" + aa[i]
a = a + "/" + "temp.txt"
file3 = open(a, 'w+', encoding='utf-8')
word = []
for line in file1:
    word = word + line.split()
ii = 0
while ii < len(res1):
    s = word[int(ii/12)]
    j = 0
    w = ""
    while j < 12:
        x = res1[j+ii] + res1[ii+j+1]
        w = w + ZWC[x]
        j = j + 2
    s1 = s + w
    file3.write(s1)
    file3.write(" ")
    ii = ii + 12
t = int(len(res1)/12)
while t < len(word):
    file3.write(word[t])
    file3.write(" ")
    t = t + 1
file3.close()
file1.close()
os.remove(file)
os.rename(a, file)
print("[INFO] ENCODING DATA Successful")
print("[INFO] LOCATION:{}".format(file))
```

```
    print("=" * 100)

def Encode():
    print("[INFO] Text Steganography ENCODING")
    print("")
    count2 = 0
    file1 = open(file, 'r')
    for line in file1:
        for word in line.split():
            count2 = count2 + 1
    file1.close()
    bt = int(count2)
    print("Maximum number of words that can be inserted:- {}".format(int(bt/6)))
    text1 = input("[*] Enter the secret message:-")
    if len(text1) > bt:
        print("[!] String is too big please reduce string size")
        Encode()
    else:
        EncodeTheText(text1)

def BinaryToDecimal(binary):
    string = int(binary, 2)
    return string

def Decode():
    print("[INFO] Text Steganography DECODING")
    ZWC_reverse = {u"\u200C": "00", u'\u202C': "01", u'\u202D': "11", u'\u200E': "10"}
    file4 = open(file, 'r', encoding="utf-8")
    temp = ''
    for line in file4:
        for word in line.split():
            T1 = word
            binary_extract = ""
            for letters in T1:
                if letters in ZWC_reverse:
                    binary_extract = binary_extract + ZWC_reverse[letters]
                if letters == "111111111111":
                    break
                else:
                    if len(binary_extract) == 12:
                        temp = temp + binary_extract
    print("[INFO] Encrypted message present in code bits: {}".format(temp))
    print("[INFO] Length of encoded bits:- {}".format(len(temp)))
    i = 0
    a = 0
    b = 4
    c = 4
    d = 12
    final = ''
    while i < len(temp):
        t3 = temp[a:b]
        a = a + 12
        b = b + 12
        i = i + 12
        t4 = temp[c:d]
        c = c + 12
        d = d + 12
        if t3 == "0110":
            for i in range(0, len(t4), 8):
                temp_data = t4[i:i + 8]
                decimal_data = BinaryToDecimal(temp_data)
                final = final + chr((decimal_data ^ 170) + 48)
        elif t3 == "0011":
            for i in range(0, len(t4), 8):
                temp_data = t4[i:i + 8]
                decimal_data = BinaryToDecimal(temp_data)
                final = final + chr((decimal_data ^ 170) - 48)
    print("[*] The Encoded data was:- {}".format(final))
    print("=" * 100)

if n == 0:
    Encode()
    else:
        Decode()
```

File video_steganography.py :

```
import math
import os
import shutil
from subprocess import call, STDOUT
import cv2
from stegano import lsb



def Video_Steganography(file, n):
    def split_string(split_str, count=10):
        per_c = math.ceil(len(split_str) / count)
        c_cout = 0
        out_str = ''
        split_list = []
        for s in split_str:
            out_str += s
            c_cout += 1
            if c_cout == per_c:
                split_list.append(out_str)
                out_str = ''
                c_cout = 0
        if c_cout != 0:
            split_list.append(out_str)
        return split_list

    def frame_extraction(video):
        if not os.path.exists("./temp"):
            os.makedirs("temp")
        temp_folder = "./temp"
        print("[INFO] temp directory is created")
        vidcap = cv2.VideoCapture(video)
        count = 0
    while True:
        success, image = vidcap.read()
        if not success:
            break
        cv2.imwrite(os.path.join(temp_folder, "{:d}.png".format(count)), image)
        count += 1

def encode_string(input_string, root="./temp/"):
    split_string_list = split_string(input_string)
    for i in range(0, len(split_string_list)):
        f_name = "{}{}.png".format(root, i)
        secret_enc = lsb.hide(f_name, split_string_list[i])
        secret_enc.save(f_name)
        print("[INFO] frame {} holds {}".format(f_name, lsb.reveal(f_name)))
    print("[INFO] The message is stored in the Embedded_Video.mp4 file")

def Decode():
    print("[INFO] Video Steganography DECODING")
    print("")
    frame_extraction(file)
    secret = []
    root = "./temp/"
    a = ''
    try:
        for i in range(0, len(os.listdir(root)) - 1):
            f_name = "{}{}.png".format(str(root), str(i))
            secret_dec = lsb.reveal(f_name)
            if secret_dec is None:
                break
            secret.append(secret_dec)
    except IndexError as e:
        print('')
    a = a.join([i for i in secret])
    print("[*] The Encoded data was:{}".format(a))
    print("")
    clean_temp()
```

```
        print("="*100)

    def clean_temp(path="./temp"):
        if os.path.exists(path):
            shutil.rmtree(path)
            print("[INFO] temp files are cleaned up")

    def Encode():
        print("[INFO] Video Steganography ENCODING")
        print("")
        input_string = input("[*] Enter the message :")
        frame_extraction(file)
        call(["ffmpeg", "-i", file, "-q:a", "0", "-map", "a", "temp/audio.mp3", "-y"], stdout=open(os.devnull,
"w"),
             stderr=STDOUT)
        encode_string(input_string)
        call(["ffmpeg", "-i", "temp/%d.png", "-vcodec", "png", "temp/Embedded_Video.mp4", "-y"],
             stdout=open(os.devnull, "w"), stderr=STDOUT)
        call(["ffmpeg", "-i", "temp/Embedded_Video.mp4", "-i", "temp/audio.mp3", "-codec", "copy",
"Embedded_Video.mp4",
              "-y"], stdout=open(os.devnull, "w"), stderr=STDOUT)
        os.remove(file)
        os.rename("Embedded_Video.mp4", file)
        clean_temp()
        print("[INFO] FILE LOCATION:{}".format(file))
        print("=" * 100)

    if n == 0:
        Encode()
    else:
        Decode()
```