ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to Dr. Roshan Chitrakar for his unwavering dedication and guidance throughout the course of the project. His passion for teaching and commitment to helping me excel in my studies have been truly inspiring.

We extend our sincere thanks to our supervisor **Er. Roshan Kumar Sah** who continuously helped us throughout the project and without his guidance, this project would have been an uphill task. We deeply express our sincere thanks to him for encouraging and allowing us to present the project on the topic "**Data Security through Cryptography and Steganography**" for the partial fulfillment of the requirements leading to the award of the degree of bachelors of engineering in information technology.

We extend our deepest appreciation to our supervisor, Er. Roshan Shah, whose mentorship and support have been instrumental in our professional growth and development. His insightful feedback, constructive criticism, and willingness to invest time in nurturing my skills have been invaluable. We deeply express our sincere thanks to him for encouraging and allowing us to present the project on the topic "**Data Security through Cryptography and Steganography**" for the partial fulfillment of the requirements leading to the award of the Degree in Bachelors of Engineering in Information Technology.

Last but certainly not lease, we want to extend profound gratitude to beloved family member, whose unwavering love and encouragement have been a pillar of strength throughout every aspect of our life.

ABSTRACT

Long-distance communication between two parties has always been vulnerable to interception. A variety of techniques, including cryptography, VPN tunneling, end-to-end encryption, and steganography, have been developed regarding security concerns for data that is being moved or stored.

Steganography is the art of concealing data on plain site. In a conventional computer file (such as an image, text file, or audio file), the unused bit is replaced by the bits of secret information using digital steganography.

Cryptography is a method of using advanced mathematical principles in storing and transmitting data in a particular form so that only those, for whom it is intended for, can read and process it.

The project, **Data Security Through Steganography and Cryptography**, will be able to encrypt and decrypt the data as well as hide the data in other images, audio files, and PDF files in plain sight, undetectable through sensory of hearing or vision.

Keywords:

Cryptography, Steganography, encrypt, decrypt

LIST OF FIGURE

TABLE OF CONTENT

1. INTRODUCTION

Data security is crucial in safeguarding sensitive information from unauthorized access, tampering, or interception. Cryptography and steganography are two distinct but complementary methods employed in information security. Cryptography involves the use of mathematical algorithms to transform plaintext into ciphertext, making it unreadable without the corresponding decryption key. Steganography, on the other hand, conceals the existence of data within seemingly innocuous carriers, thereby enabling covert communication and secure data transfer.

Our project, Data Security through Steganography and cryptography, encrypts information provided by encryption algorithms and hides this cypher text in other media like image, audio and video files.

1.1. BACKGROUND STUDY

The roots of cryptography can be traced back to ancient times, where primitive methods were used to conceal messages. The advent of computers and the digital era revolutionized cryptography, leading to the development of symmetric and asymmetric encryption algorithms such as DES, RSA, and AES. Public key cryptography introduced a breakthrough in secure communication, allowing parties to exchange encrypted data without a shared secret key.

Steganography operates on the premise of hiding sensitive data within seemingly innocuous carriers, such as images, audio files, or text. The study examines various steganographic techniques, including Least Significant Bit (LSB) embedding, spread spectrum, and transform domain techniques. It also evaluates the capacity, security, and detection challenges associated with different steganographic methods.

The integration of cryptography and steganography can significantly enhance data security. This section explores hybrid approaches that combine encryption and steganography to create robust and covert communication channels. Such methods offer double-layered protection, where encryption secures the payload and steganography conceals its existence.

1.2. PROBLEM STATEMENT

The rapid development of digital media technology has been driven by the need for a quick and effective means of storing and sharing information. The technology at our time has given us a time and space-efficient way to store and transfer those data, but the data are still vulnerable to attack, directly on the host or while being sent. As a result, we needed to take a number of security measures, such as access restrictions, VPN tunneling, steganography, and cryptography.

Cryptography has been dated to be in use even before the Egyptian civilization and we still use the technique. So, it is no doubt that cryptography is the best tool to conseal an information. However, Cryptography only hides the message, which works most of the time. But with the right key, one can easily decrypt the encrypted message. One of the best measure that could be taken to prevent unauthorized access it to hide the encrypted data such that its existence itself is hidden. This is where Steganography comes in.

Integrating these two, cryptography and steganography, is the way I chose to ensure that the application is robust. Therefore, this system will be able to maintain the security of the information being transmitted.

1.3. PROJECT OBJECTIVES

We aim to acheive data security by encrypting the data and hiding its existence in plain site. In order to acheive this, here are some core objective of the project

1. Encrypting the data through cryptography.
2. After encryption, the file is then embedded into other media format using steganography. This adds another layer of security to the data by hiding it in plain site.
3. Finally, data in the carrier media is then extracted to get the embedded file from carrier media.

1.4. SCOPE AND IMPORTANCE

Data security through steganography and cryptography encompass the protection of sensitive information, ensuring confidentiality, integrity, and authenticity.

1. Steganography allows one to exchange sensitive information without attracting attention.

2. It can also be used to securely store confidential data by hiding it within seemingly innocuous files.
3. Implementing both steganography and cryptography ensures that sensitive information remains confidential and accessible only to authorized parties.

By implementing strong data security measures, including steganography and cryptography, organizations can demonstrate their commitment to safeguarding sensitive information.

1.5. LIMITATION

The Limitation of the project are:

1. This project only allows .wav audio format.
2. Only lossless image file format could be used like .png.
3. Being a standalone desktop application, the exchange of the password to decrypt should be done through another secure channel.

2. LITERATURE STUDY

In the digital age, data security has become a critical concern as sensitive information is transmitted and stored electronically. Cryptography and steganography are two fundamental techniques used to protect data from unauthorized access and ensure its confidentiality, integrity, and authenticity. This literature review aims to explore existing research and studies related to data security through cryptography and steganography, their principles, applications, and the challenges they address in ensuring robust data protection.

The best measure to protect a confidential data is to conceal its existence. But just by consealing its existence does not ensure complete security. So before consealing, we encrypt the data using a robust encryption algorithmusing a encryption algorithm. This is done such that a third-party that may get hold of the information could not figure out the content of the data. Thus this project will implement the security practice of Steganography and cryptography.

Using cryptography data is first encrypted into a format that is unreadable by an unauthorized user. Once we have the cypher text, the data is then embedded into an image, audio or a video file so as not to arouse suspicion from an eavesdropper.

This Project intend to provide user with a symmetric encryption algorithm like AES-256.

2.1. EVOLUTION OF CRYPTOGRAPHY

The historical roots of cryptography can be traced back to ancient civilizations, where simple methods like substitution ciphers were employed to hide messages. Over the years, the field of cryptography has evolved significantly, with the advent of computers enabling the development of complex encryption algorithms. Research has focused on historical developments, providing insights into how cryptographic techniques have evolved and how modern algorithms build upon traditional methods (Schneier, 1996).

2.2. CRYPTOGRAPHY ALGOTITHM AND TECHNIQUES

This section delves into the various cryptographic algorithms and techniques used to secure data. Research papers have extensively explored symmetric encryption algorithms like Data Encryption Standard (DES) and Advanced Encryption Standard (AES), as well as asymmetric algorithms like RSA and Elliptic Curve Cryptography (ECC). Studies have compared the performance, security, and computational efficiency of different encryption methods (Paar & Pelzl, 2010).

2.3. MODERN APPLICATIONS OF CRYPTOGRAPH

This section explores the practical applications of cryptography in securing data across various domains. Studies have examined its role in securing communications through Virtual Private Networks (VPNs), secure messaging applications, and email encryption (Johnson, 2001). Moreover, cryptographic techniques are applied to secure data stored in cloud environments, ensuring data privacy and confidentiality (Ristenpart et al., 2009).

2.4. FUNDAMENTALS OF STEGANOGRAPHY

Research in steganography has grown substantially, focusing on the methods used to embed secret data within innocuous carriers. Studies have examined the capacity, security, and detection challenges associated with various steganographic techniques, including Least Significant Bit (LSB) embedding, spread spectrum, and transform domain techniques (Westfeld & Pfitzmann, 2000).

2.5. CRYPTOGRAPHY AND STEGANOGRAPHY INTEGRATION

Integration of cryptography and steganography has gained attention as a means to enhance data security. Research has explored hybrid approaches, where encryption secures the payload, and steganography conceals its existence (Barni et al., 2001). These integrated techniques offer double-layered protection and have applications in secure data communication and storage.

2.6. CHALLENGES AND FUTURE DIRECTIONS

Researchers have highlighted various challenges in data security through cryptography and steganography. As computing power increases, studies have addressed the need for post-quantum cryptography to withstand quantum computing threats (Dworkin, 2016). Additionally, advancements in artificial intelligence and machine learning have prompted investigations into adversarial attacks on cryptographic systems (Carlini & Wagner, 2018). Future research directions include exploring more robust steganographic methods and studying the implications of emerging technologies on data security.

2.7. EXISTING SYSTEM

2.7.1. OpenegSto

OpenStego perform Steganography effectively with image files of type JPEG, JPG, BMP, GIF, PNG etc. The output of OpenStego is a PNG file. It is an open source and free Steganography tool developed using Java. It also provide watermarking which is used to detect an unauthorized copy of image files. But it does not support audio steganography and encryption of the files.

2.7.2. RSteg

RSteg is also image Steganography tool developed using Java. Performing Steganography using RSteg is simple. All that is require is an Image file, text to be encrypted and password to be set for decryption. The final output is stored as PNG. The stegano-image plug into the same Steganography detection tool for decryption along with a password.

Although similar kind of software are available to perform the steganography on both images and audio steganography, there is no any software that are equipped with all the feature like steganography using audio, image and video files with additional encryption technique. Thus this project of our tried to bring together all this features in a single desktop program.

3. METHODOLOGY

3.1. SYSTEM DESIGN

To designed the system we used LSB to embed the data to the carrier file for steganography, and AES encryption for cryptography.

3.1.1. BASIC STRUCTURE OF STEGANOGRAPHY

The basic structure of Steganography is made up of three components.

1. **Carrier** - The carrier can be a painting, a digital image, audio file, even a TCP/IP packet among other things. It is the object that will 'carry' the hidden message.
2. **Message** - The message (hidden) is being carried by the object (carrier).
3. **Password** - A key is used to decode/decipher/discover the hidden message.

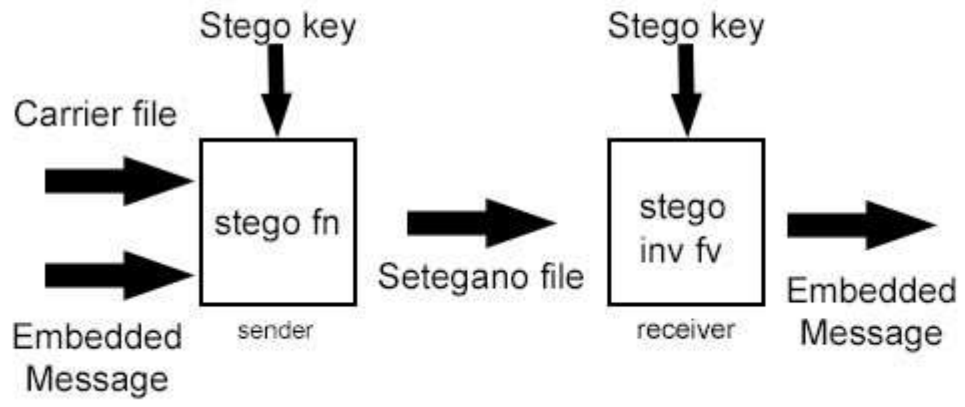


Figure 1: Basic Steganograph structure

Here, the carrier media is given as input along with the file that is to be hidden in that media. The file is hidden into the carrier media using the steganographic algorithm i.e. LSB algorithm in this project. In stegano analysis, the message or the hidden file is extracted from the carrier media.

3.1.2. CRYPTOGRAPHY VS STEGANOGRAPHY

Steganography is not the same as cryptography. Cryptography hides the contents of a secret message from malicious people, whereas steganography conceals the existence of the message. In cryptography, the structure of a message is scrambled to make it meaningless and unattenable unless the decryption key is available. In contrast, steganography does not alter the structure of the secret message, but hides it inside a cover media so it cannot be seen.

3.1.3. STEGANOGRAPHY TECHNIQUES:

Digital data can be embedded in many ways into the carrier media, either using a spatial domain (LSB Replacement, Matrix embedding, Histogram modification) or transform domain (Discrete cosine Transform, Fast Fourier Transform) steganographic algorithm, the message is hidden into the cover media and the stegno media is obtained. The most common techniques of data hiding in images are:

1. **Appending data bytes at the end of carrier:** The secret data bytes are appended at the end of the carrier media such as image and the carrier media is then compressed to its original size to reduce the suspects of having secret data. Advantage is that it is very easy to implement. Disadvantage is it is very easy to detect and get the message.
2. **Transform domain based embedding**: Transform Embedding Techniques embed the data by modulating coefficients in a transform domain, such as Discrete Fourier Transform (DFT), Discrete Cosine Transform (DCT) (used in JPEG compression) or Discrete Wavelet Transform (DWT). Modifying the transform coefficients provides more robustness to the compression (especially to lossy), cropping, or some image processing, than LSB techniques.
3. **Least significant bit (LSB) insertion:** LSB techniques embed the message bits directly into the least significant bit plane of the cover image in a deterministic sequence. Here the binary representations of the secret data have been taken and the LSB of each byte is overwritten. This results in a change with too low amplitude to be human-perceptible. LSB embedding is simple, popular and many use these technique.

3.1.4. LSB ENCODING ALGORITHM

1. Carrier media and the Message file along with the key is inputted.
2. Convert the message file into the binary format and generate the stream of bits
3. Bytes representing the carrier-media is taken in a single array and byte stream is generated.
4. Message bits are taken sequentially and then are placed in LSB of the byte representing the carrier media

5. Repeat the step 4. Till all the message bits are placed in the Least LSB.

Output: stegno-media

3.1.5. LSB DECODING ALGORITHM

1. The Stegano-media and the key is inputted.
2. Array of the bytes are generated.
3. The total number of bits of message and the bytes representing the stego- media are taken.
4. The bits stream of the message is generated.
6. Available bits are grouped to form bytes such that each byte represents single ASCII character.
7. Character are stored in the text file.

Output: Recovered hidden message text file.

3.1.6. ADVANCED ENCRYPTION STANDARD

The Advanced **Encryption Standard** (**AES**), also known by its original name **Rijndael** is a specification for the encryption of electronic data established by the US. National Institute of Standard and Technology in 2001. [*Wikipedia*].

AES is widely adopted symmetric encryption algorithm after its declassified .In 2003, The US government begin to use AES as the encryption standard for protecting classified information. AES is a symmetric key symmetric block cipher. It comprises three block ciphers: AES-128, AES-192, and AES-256. Each cipher encrypts and decrypts data in the block of 128-bits using the cryptographic keys of 128, 192 or 256 bits respectively.

3.1.6.1. OPERATION OF AES

AES creates an iterative cipher. It is based on the substitution-permutation network. It comprises of a series of linked operation, some of which involve replacing inputs by specific outputs (substitution) and permutation.

AES performs all its computations on bytes rather than bits. The algorithm treats the 128 bits of a plain text block as 16 bytes. These 16 bytes are arranged in four columns and four rows for processing as a matrix.

The AES encryption algorithm defines a number of transformations that are to be performed on data stored in an array. The first step of the cipher is to put the data into an array; after which the cipher transformations are repeated over a number of encryption rounds. AES uses 10 rounds for 128 bit keys, 12 rounds for 192 bit keys and 14 rounds for 256 bit keys. Each of these rounds uses different 128-bit round key which is calculated from the original AES key.

3.1.6.2. ENCRYPTION PROCESS

The AES encryption algorithm defines a number of transformations that are to be performed on data stored in an array. The first step of the cipher is to put the data into an array; after which the cipher transformations are repeated over a number of encryption rounds. AES uses 10 rounds for 128 bit keys, 12 rounds for 192 bit keys and 14 rounds for 256 bit keys. Each of these rounds uses different 128-bit round key which is calculated from the original AES key.

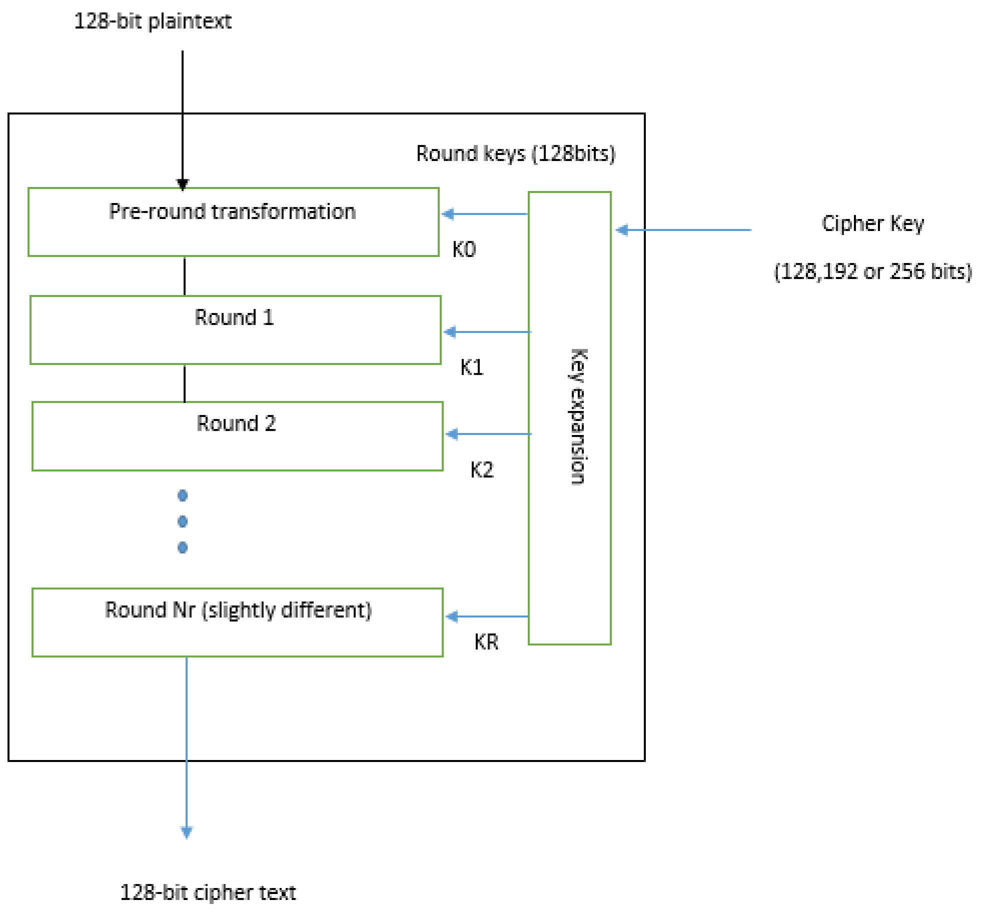


Figure 2:AES encryption

AES consists of four different types of layers, each of them manipulates all 128 bits of the data path (also called states).

Each round, with the exception of the first, consists of all three layers and they are

Bytes Substitution layer: The 16 input bytes are substituted by looking up a fixed table (S box) given in design. The result is in a matrix of four rows and four columns

Shift Rows: Each of the four rows of the matrix is shifted to the left.

Mix columns: Here this function takes input the four byte of one column and outputs four completely new bytes which replaces the original column. This step is not performed in the last round

Add round key: The 16 bytes of matrix are now considered as 128 bits and are XORed to the 129bits of the round key. If this is the last round then the output is the cipher text. Otherwise the resulting 128bits are interpreted as 16 byte and we begin another similar round.

3.1.6.3. DECRYPTION PROCESS

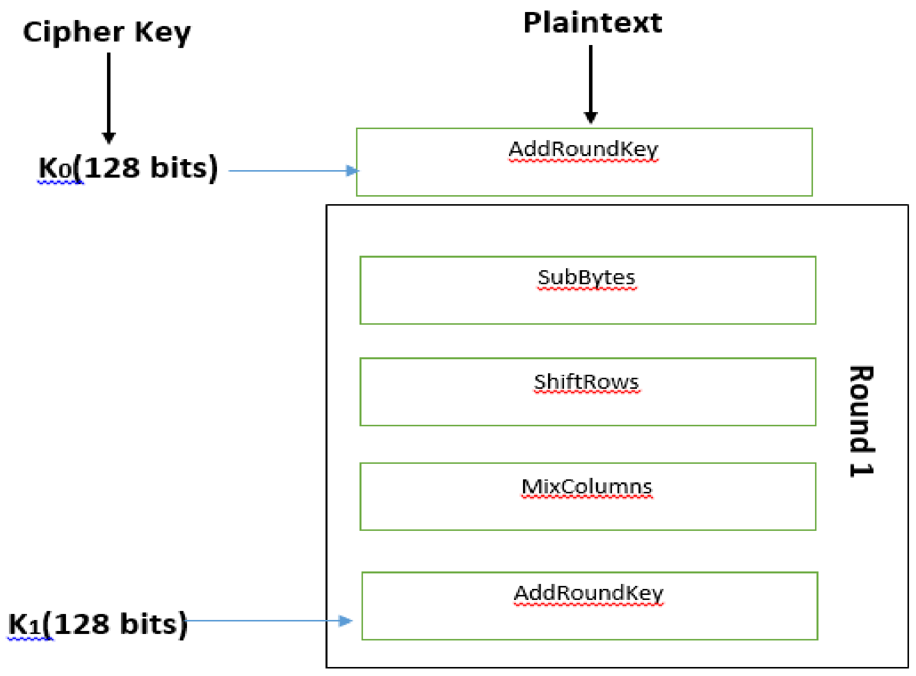


Figure 3: AES encryption 2

The Process of decryption of an AES cipher text is similar to the encryption process in the reverse order. Each round consists of the four processes conducted in the reverse order.

1. Add round key
2. Mix columns
3. Shift rows
4. Byte substitution

3.1.7. BLOCK CIPHER AND MODES OF OPERATION

A block cipher processes the data block of fixed size. Usually the size of a message is larger than the block size. Hence, the long message is divided into a series of sequential message blocks, and the cipher operates on these blocks one at a time. The strength of the encryption scheme depend upon the key length not the block size.

MODES OF OPERATION

There are different modes of the operation of a block cipher. The different modes result in the different properties being achieved which add to the security of the underlying block cipher. Some of them are:

EBC: Electronic Code Book

CBC: Cipher Block Chaining

CFC: Cipher Feedback Mode

CIPHER CHAINING MODE

CBC mode of operation provides message dependence for generating cipher text and makes the system non-deterministic. In CBC mode, the current plaintext block is added to the previous cipher text block, the result is encrypted with the key. Decryption is thus the reverse process, which involves decrypting the current cipher text and then adding the previous cipher text block to the result.

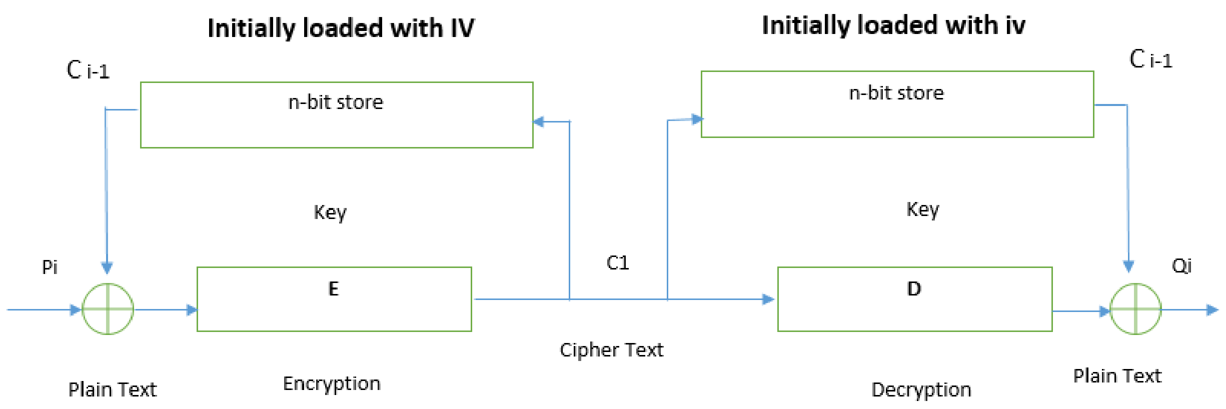


Figure 4: Cypher Chaining Mode

3.1.8. Use case

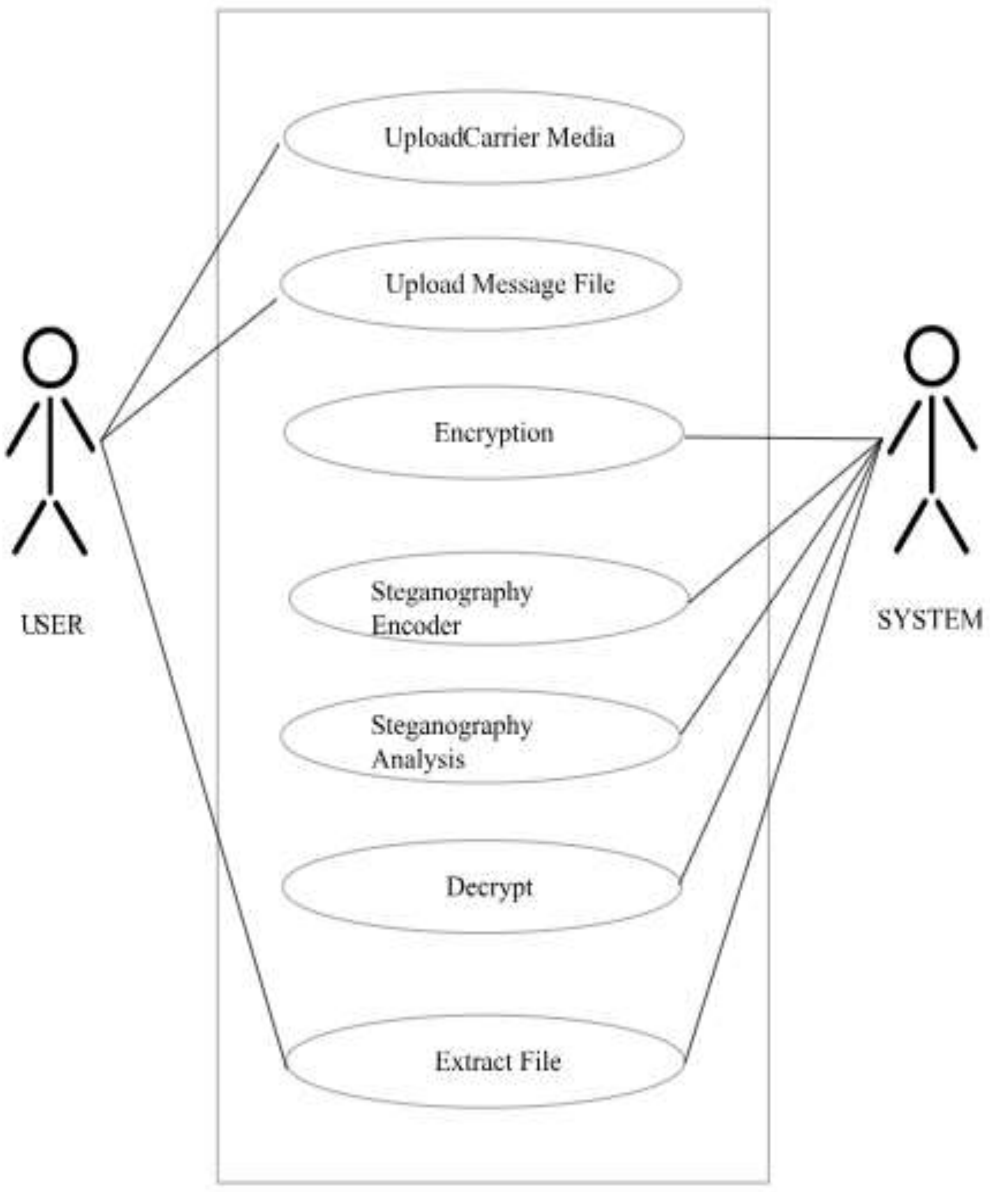


Figure 5:Use Case Diagram

3.1.9. Flowchart

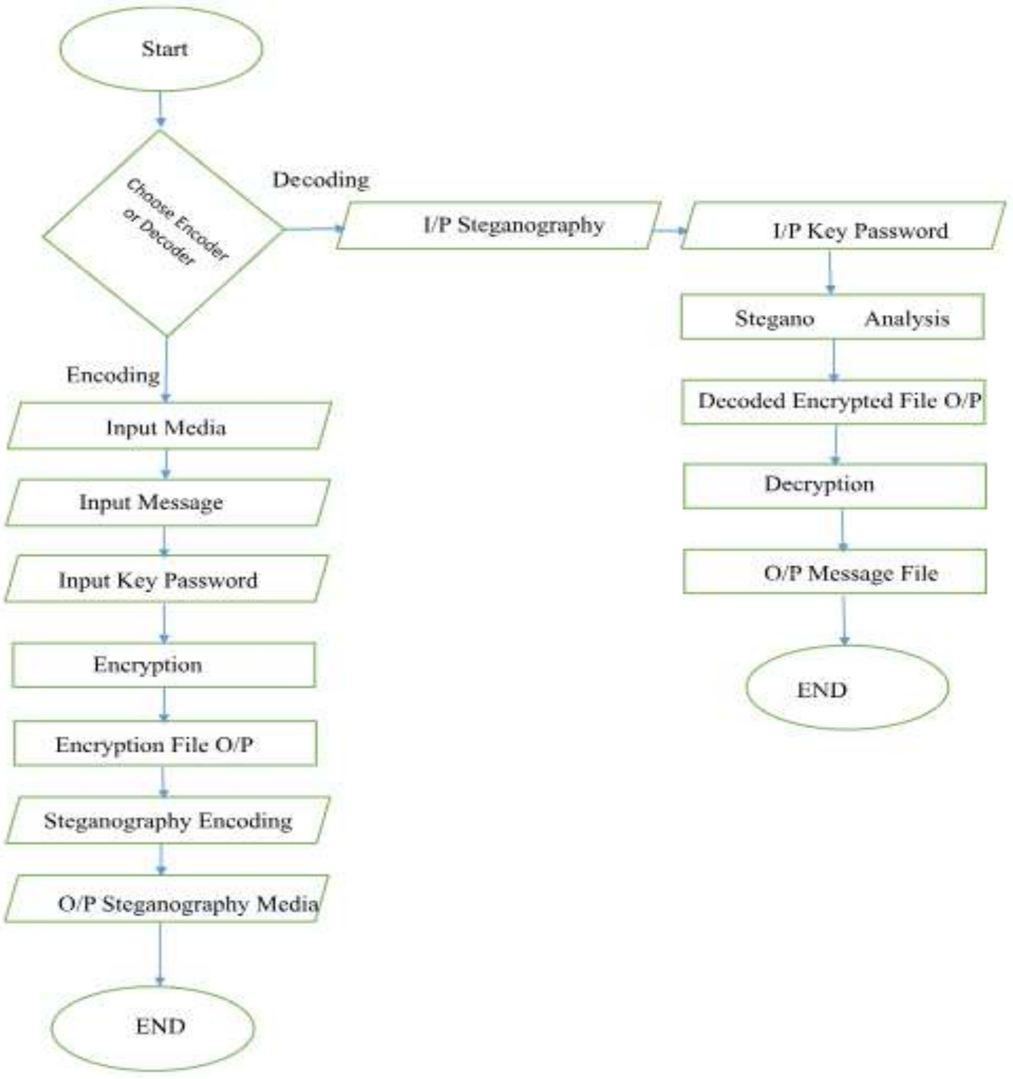


Figure 6: Flow chart of the system

3.1.10. System Sequence Diagram

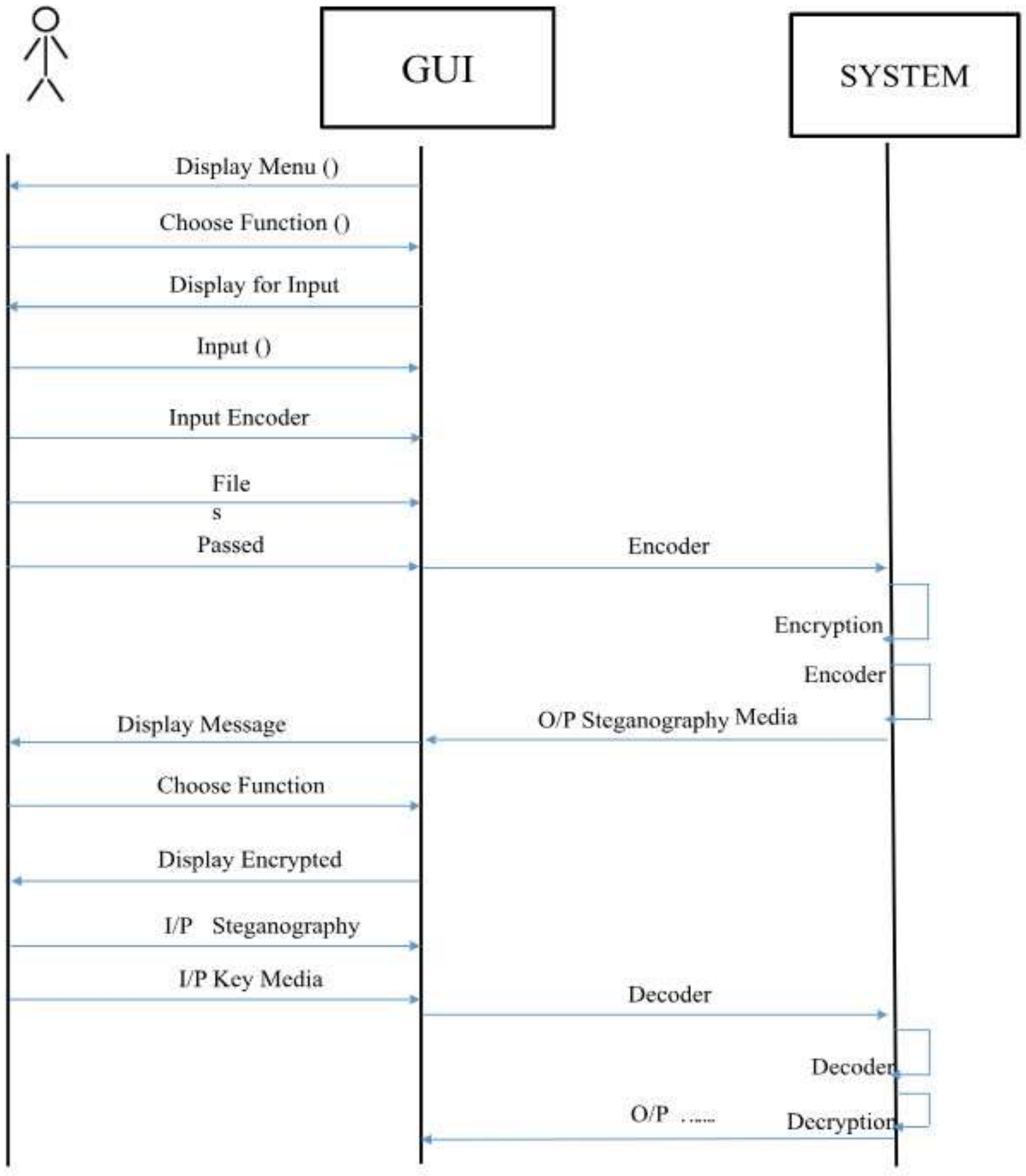


Figure 7: System sequence diagram

3.2. Method

For the development of this project we choose to use Incremental Model as this model provide an easy and effective way to work on one function at a time and gradually completing the all other function to give final fully- functional software.

3.3. Model

3.3.1. Incremental Model

The framework we will be using for developing this project is Incremental Model. This model combines linear sequential model with the iterative prototype model. New functionalities will be added as each increment is developed. The phases of linear sequential model are: Analysis, Design, Coding and Testing. The software repeatedly passes through these phase in iteration and an increment is delivered with progressive changes. Total of 6 increment is expected to be required to complete the project.

- 1st Iteration: Encryption and decryption of text file.
- 2nd Iteration: Encryption of data into .PNG file.
- 3rd Iteration: Encryption of data into audio (.wav) file.
- 4th Iteration: Decryption of data from both .PNG and .WAV file
- 5th Iteration: Encryption and decryption of text file into .MOV file
- 6th Iteration: Graphical User Interface and implementation of all iteration to a single Program.

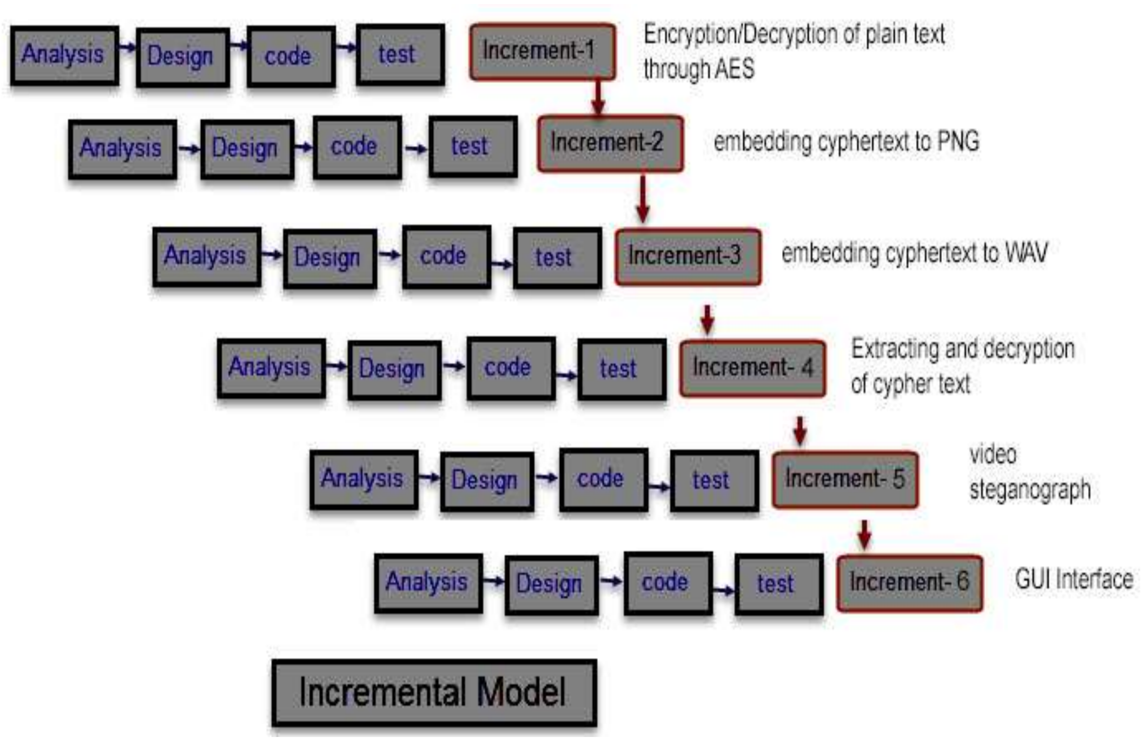

Analysis
Design
code
test
Increment-1
Encryption/Decryption of plain text through AES
Increment-2
embedding cyphertext to PNG
Increment-3
embedding cyphertext to WAV
Increment- 4
Extracting and decryption of cypher text
Increment- 5
video steganograph
Increment- 6
GUI Interface
Incremental Model

Figure 8: Incremental Model

3.3.1.1. Analysis Phase

In this phase, requirement analysis was performed in order to find out the requirements of the system. The outcome of this phase is a SRS which is an acronym for "System Requirement Specifications".

3.3.1.2. Design phase

In this phase, the System Requirement Specification is translated into the system's design. Here, Entity Relationship Diagram, Use Case Diagram, Flowcharts was developed.

3.3.1.3. Coding Phase

During this phase we implement our design using the python code. The programs are coded and each feature run as a indivudual mode. Intergration of these module is left to be done.

3.3.1.4. Testing Phase

During this phase Unit test on each of the four module has been successfully performed.

3.4. Requirement Analysis

Requirements analysis encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product or project, taking account of the possibly conflicting requirements of the various stakeholders, analyzing, documenting, validating and managing software or system requirements [wiki]. Analysis phase emphasizes an investigation of the problem and the requirement rather than a solution.

3.4.1. Input Requirement

The input requirement of this project are:-

- A carrier media
- A text file that has a message
- A passphrase to generate the key

3.4.2. Output Requirement

Since the output of the program yield the steganomedia. Output requirement for the program is directory where we can create and write a file.

3.4.3. Functional Requirement

- Encrypt the message file using the encryption algorithm
- Decrypt the encrypted file using the decryption algorithm.
- Embed the encrypted file into the carrier media
- Decode the embedded encrypted file into the plain text file.

3.4.4. Interface Requirement

- Provide user to select the image and message file from the storage.
- Provide user the interface to input the password and the key.

3.5. Testing

Using different test cases, test for each of the function developed until this period has been performed.

3.5.1. Testing table

Test No	Function	Test	Expected Result	Outcome	Evidence
1.	Encryption	Generate the cipher text from the message file	A encrypted file containing the cipher text	Successful	Appendix B
2.	Decryption	Decrypt the encrypted file	A text file with readable message	Successful	Appendix B
3.	Encoding	Embed the message file in image	The message hidden in the image	Successful	Appendix C
4.	Decoding	Decode carrier image to get hidden file	Recovered text file	Successful	Appendix C
5.	Encoding	Embed the message file in audio file	The message file hidden in the audio	Successful	Appendix D
6.	Decoding	Decode carrier audio to get hidden file	Recovered text file	Successful	Appendix D
7.	Encoding	Encode the message file in video file	The message file hidden in the video	Successful	
8	Decoding	Decode the carrier video file to get hidden message	Recovered text file	Successful	

4. FURTHER WORK

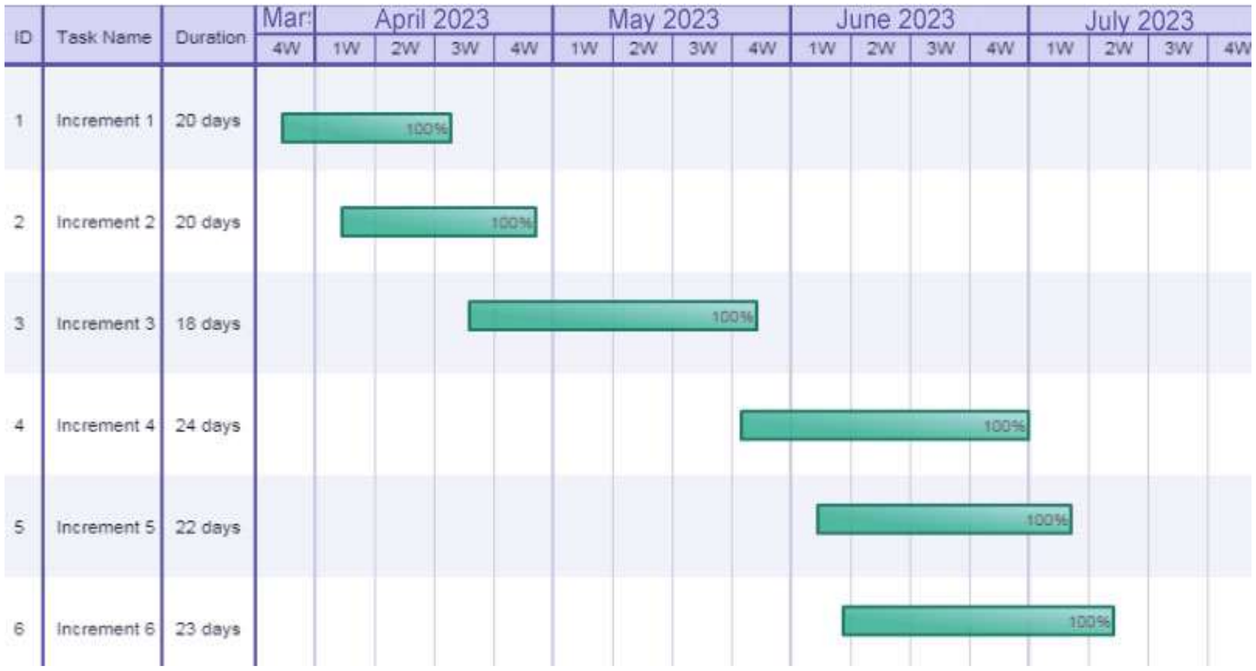
- Our system is a standalone program but the same technique could be implemented for multiple machine over the network
- A public key cryptography technique could be introduced to make the system effective.
- Support to the more types of carrier media could be added i.e .jpeg , mp3, flv, etc.
- The UI of the application could be better and more user friendly

5. TASK AND TIME SCHEDULE

5.1. Time Schedule

Phases	**1st Increment**	**2nd Increment**	**3rd Increment**	**4th Increment**	**5th Increment**	**6th Increment**
Analysis	5 Days	5 Days	3 Days	3 Days	3 Days	5 Days
Design	5 Days	5 Days	5 Days	5 Days	5 Days	5 Days
Coding and Implementation	7 Days	7 Days	7 Days	10 Days	10 Days	7 Days
Testing and Debugging	2 Days	2 Days	2 Days	2 Days	2 Days	2 Days
Documentation	1 Day	1 Day	1 Day	4Days	2Days	4Days
Total Duration	**20 Days**	**20 Days**	**18 Days**	**24 Days**	**22days**	**23 Days**

5.2. Gantt Chart



ID	Task Name	Duration
1	Increment 1	20 days
2	Increment 2	20 days
3	Increment 3	18 days
4	Increment 4	24 days
5	Increment 5	22 days
6	Increment 6	23 days

Figure 9: Gantt Chart

6. CONCLUSION

This project has proved that Steganography along with the cryptography help us to obtain the data security of confidential message during communication through less secure communication medium.

It is observed that through LSB substitution steganographic method, the result obtained in data hiding are pretty impressive. The implementation of this method is very simple and effective. Even though this program does not support the lossy compression file format it support wide range of the carrier media like image, audio video and pdf.

Here to make the data more secure, data is encrypted using the AES-256 encryption algorithm. A password is need for the encryption and the decryption of the message file. Sharing the password is to be done using a secure communication medium or agreed upon beforehand.

APPENDIX

Code Snippets

Advanced Encryption System

```
import os, random, string
from Crypto.Cipher import AES
from Crypto.Hash import SHA256
from Crypto import Random
def GenerateKey():
     key = ''.join(random.choices(string.ascii_uppercase + string.ascii_lowercase + 
string.digits, k=32))
    return key.encode('utf-8')
def Encrypt(key, filename):
    print("Encrypting.......")
    chunksize = 64* 1024
    outputFile = filename +".hidn"
    filesize = str(os.path.getsize(filename)).zfill(16)
    IV = Random.new().read(16
    encryptor = AES.new(key, AES.MODE_CBC, IV)
    with open(filename, 'rb') as infile:
        with open(outputFile, 'wb') as outfile:
            outfile.write(filesize.encode('utf-8'))
            outfile.write(IV)
            while True:
                chunk = infile.read(chunksize)
                if len(chunk) == 0:
                    break
                elif len(chunk) % 16 != 0:
                    chunk += b' ' * (16 - (len(chunk) % 16))
                outfile.write(encryptor.encrypt(chunk))
    os.remove(filename)
def Decrypt(key, filename):
    print("Decrypting.......")
    chunksize = 64*1024
    outputFile = filename[:-5]
    with open(filename, 'rb') as infile:
        filesize = int(infile.read(16))
        IV = infile.read(16)
        decryptor = AES.new(key, AES.MODE_CBC, IV)
        with open(outputFile, 'wb') as outfile:
            while True:
                chunk = infile.read(chunksize)
                if len(chunk) == 0:
                    break
                outfile.write(decryptor.decrypt(chunk))
            outfile.truncate(filesize)
    os.remove(filename)
def GetKey(password):
    hasher = SHA256.new(password.encode('utf-8'))
    return hasher.digest()
def Test():
    userInputEncryptDecrypt = input("Press 1 for Encrypt and 2 for Decrypt: ")
    if userInputEncryptDecrypt == '1':
        filename = input("Type in File that you want to encrypt: ")
        password = input("Choose a Password: ")
        Encrypt(GetKey(password), filename)
        print("Completed")
    elif userInputEncryptDecrypt == '2':
        filename = input("Type in Filename that you want to decrypt: ")
        password = input("Type in the Password: ")
        Decrypt(GetKey(password), filename)
        print("Completed")
    else:
        print("Either Select 1 or 2")
if __name__ == '__main__':
    Test()
```

Image Steganography

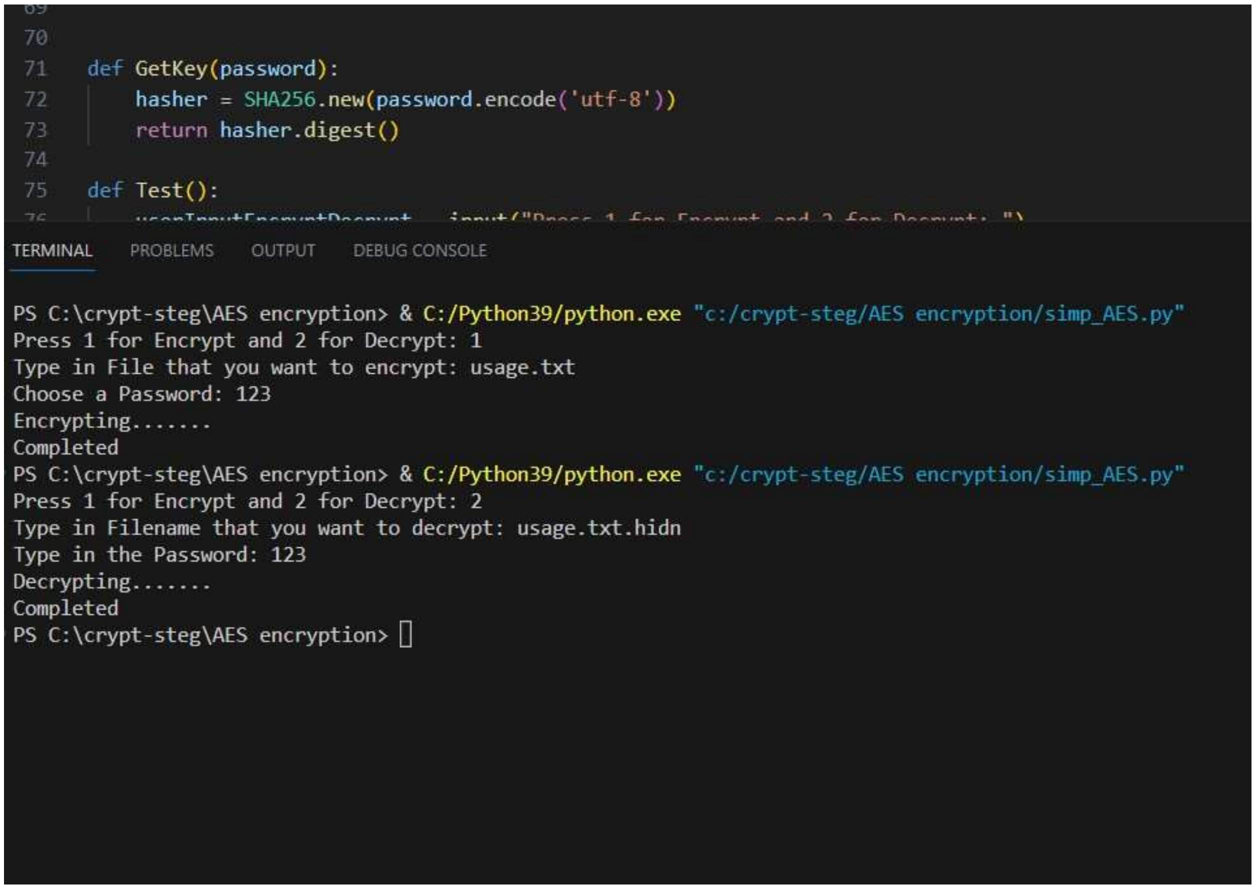
```
import cv2
import docopt
import numpy as np
from simp_AES import *
class SteganographyException(Exception):
    pass
class LSBSteg():
    def __init__(self, im):
        self.image = im
        self.height, self.width, self.nbchannels = im.shape
        self.size = self.width * self.height
        self.maskONEValues = [1,2,4,8,16,32,64,128]
        self.maskONE = self.maskONEValues.pop(0)
        self.maskZEROValues = [254,253,251,247,239,223,191,127]
        self.maskZERO = self.maskZEROValues.pop(0)
        self.curwidth = 0
        self.curheight = 0
        self.curchan = 0
    def put_binary_value(self, bits):
        for c in bits:
            val = list(self.image[self.curheight,self.curwidth])
            if int(c) == 1:
                val[self.curchan] = int(val[self.curchan]) | self.maskONE
            else:
                val[self.curchan] = int(val[self.curchan]) & self.maskZERO
            self.image[self.curheight,self.curwidth] = tuple(val)
            self.next_slot()
    def next_slot(self):
        if self.curchan == self.nbchannels-1:
            self.curchan = 0
            if self.curwidth == self.width-1: of the same line
                self.curwidth = 0
                if self.curheight == self.height-1
                    self.curheight = 0
                    if self.maskONE == 128:
                        raise SteganographyException("No available slot remaining
(image filled)")
                    else:
                        self.maskONE = self.maskONEValues.pop(0)
                        self.maskZERO = self.maskZEROValues.pop(0)
                else:
                    self.curheight +=1
            else:
                self.curwidth +=1
        else:
            self.curchan +=1
    def read_bit(self):
        val = self.image[self.curheight,self.curwidth][self.curchan]
        val = int(val) & self.maskONE
        self.next_slot()
        if val > 0:
            return "1"
        else:
            return "0"
    def read_byte(self):
        return self.read_bits(8)
    def read_bits(self, nb):
        bits = ""
        for i in range(nb):
            bits += self.read_bit()
        return bits
    def byteValue(self, val):
        return self.binary_value(val, 8)
    def binary_value(self, val, bitsize):
        binval = bin(val)[2:]
        if len(binval) > bitsize:
            raise SteganographyException("binary value larger than the expected size")
        while len(binval) < bitsize:
            binval = "0"+binval
        return binval
    def encode_binary(self, data):
        l = len(data)
        if self.width*self.height*self.nbchannels < l+64:
            raise SteganographyException("Carrier image not big enough to hold all the
datas to steganography")
        self.put_binary_value(self.binary_value(l, 64))
        for byte in data:
            byte = byte if isinstance(byte, int) else ord(byte)
            self.put_binary_value(self.byteValue(byte))
        return self.image
    def decode_binary(self):
        l = int(self.read_bits(64), 2)
        output = b""
        for i in range(l):
            output += chr(int(self.read_byte(),2)).encode("utf-8")
        return output
def main():
    args = docopt.docopt(__doc__, version="0.2")
    in_f = args["--in"]
    out_f = args["--out"]
    in_img = cv2.imread(in_f)
    steg = LSBSteg(in_img)
    if args['encode']:
        filename = args["--file"]
        password = args["--pass"]
        data = open(filename , "rb").read()
        Encrypt(GetKey(password), filename)
        res = steg.encode_binary(data)
        cv2.imwrite(out_f, res)
    elif args["decode"]:
        password = args["--pass"]
        raw = steg.decode_binary()
        with open(out_f, "wb") as f:
            f.write(raw)
if __name__=="__main__":
    main()
```

Graphical User Interface

```
import sys
from tkinter import *
from tkinter.filedialog import askopenfilename
import os.path
root = Tk()
leftFrame = Frame(root,relief=RIDGE,borderwidth=2,width=250,height=500)
leftFrame.grid(row = 0,column =0) #frame is put at the left
rightFrame = Frame(root,relief=RIDGE,borderwidth=2,width=250,height=500
global cfile,msgfile, stegfile, passwd,oFileget,bytes_to_recover
oFileg = StringVar()
passwdd = StringVar()
bytorecv = StringVar()
def opencmediafile():
    global cfile
    cfile = askopenfilename()
def openmsgfile():
    global msgfile
    msgfile = askopenfilename()
def stegmedia():
    global stegfile
    stegfile = askopenfilename()
def hide():
    global cfile,msgfile,stegfile,passwd,oFileget
    oFileget = oFileg.get()
    passwd = passwdd.get()
    ext = os.path.splitext(cfile)[1]
    if ext == '.png':
        script = 'python3 LSBSteg.py encode -i ' + cfile +' -o ' + oFileget + ' -f '
+ msgfile + ' -p ' + passwd
        os.system(script)
    elif ext == '.wav':
        script = 'python3 wav-steg.py -h -d '+ msgfile + ' -s ' + cfile + ' -o ' +
oFileget + ' -p ' + passwd
        os.system(script)
    elif ext == '.mov':
        script= 'python3 videosteg.py encode -i ' + cfile + ' -o ' + oFileget + ' -f
' + msgfile + ' -p ' + passwd
        os.system(script)
    else:
        exit()
def unhide():
    global stegfile,passwd,oFileget,bytes_to_recover
    oFileget = oFileg.get()
    passwd = passwdd.get()
    bytes_to_recover=bytorecv.get()
    ext = os.path.splitext(stegfile)[1]
    if ext == '.png':
        script = 'python3 LSBSteg.py decode -i ' + stegfile +' -o ' + oFileget +  ' -
p ' + passwd
        os.system(script)
    elif ext == '.wav':
        script = 'python3 wav-steg.py -r -s ' + stegfile + ' -o ' + oFileget + ' -p '
+ passwd + ' -b ' + bytes_to_recover
        os.system(script)
    elif ext == '.mov' :
        script= 'python3 videosteg.py decode -i ' + stegfile + ' -o ' + oFileget + '
-p ' + passwd
        os.system(script)
    else:
        exit()
#Encode format
Luploadmedia = Label(leftFrame,text="Upload Media")
Luploadmsg = Label(leftFrame,text="Upload message file")
Lpasswd = Label(leftFrame,text="Password")
Loutfilename = Label(leftFrame,text="Outputfilename:")
carriermedia = Button(leftFrame,text="upload",width=20,command=opencmediafile)
messagefile = Button(leftFrame,text="upload",width=20,command =openmsgfile)
oFile = Entry(leftFrame,textvariable=oFileg)
password = Entry(leftFrame,textvariable=passwdd)
hideButton = Button(leftFrame,text="HIDE",width=20,command=hide)
Luploadmedia.grid(row=1,column=1,sticky=E)
Luploadmsg.grid(row=3,column=1,sticky=E)
Lpasswd.grid(row=5,column=1,sticky=E)
```

```
Loutfilename.grid(row=7,column=1,sticky=E)
carriermedia.grid(row=1,column=2)
messagefile.grid(row=3 ,column=2)
password.grid(row=5,column=2)
oFile.grid(row=7,column=2)
hideButton.grid(row=9,column=3)
#Decode format
Luploadstegmedia = Label(rightFrame,text="Steg Media")
Lpasswd = Label(rightFrame,text="Password")
Loutfilename = Label(rightFrame,text="Outputfilen:")
Lbytestorecover = Label(rightFrame,text="Bytes to recover(for audio:)")
stegmedia =  Button(rightFrame,text="upload",width=20,command=stegmedia)
oFile = Entry(rightFrame,textvariable=oFileg)
password = Entry(rightFrame,textvariable=passwdd)
Btorecover = Entry(rightFrame,textvariable=bytorecv)
unhideButton = Button(rightFrame,text="UNHIDE",width=20,command=unhide)
Luploadstegmedia.grid(row=1,column=1,sticky=E)
Lpasswd.grid(row=3,column=1,sticky=E)
Loutfilename.grid(row=5,column=1,sticky=E)
stegmedia.grid(row=1,column=2)
password.grid(row=3,column=2)
oFile.grid(row=5,column=2)
unhideButton.grid(row=9,column=3)
Btorecover.grid(row=7,column=2)
Lbytestorecover.grid(row=7,column=1,sticky=E)
root.mainloop()
```

Appendix A: Encryption/Decryption Test



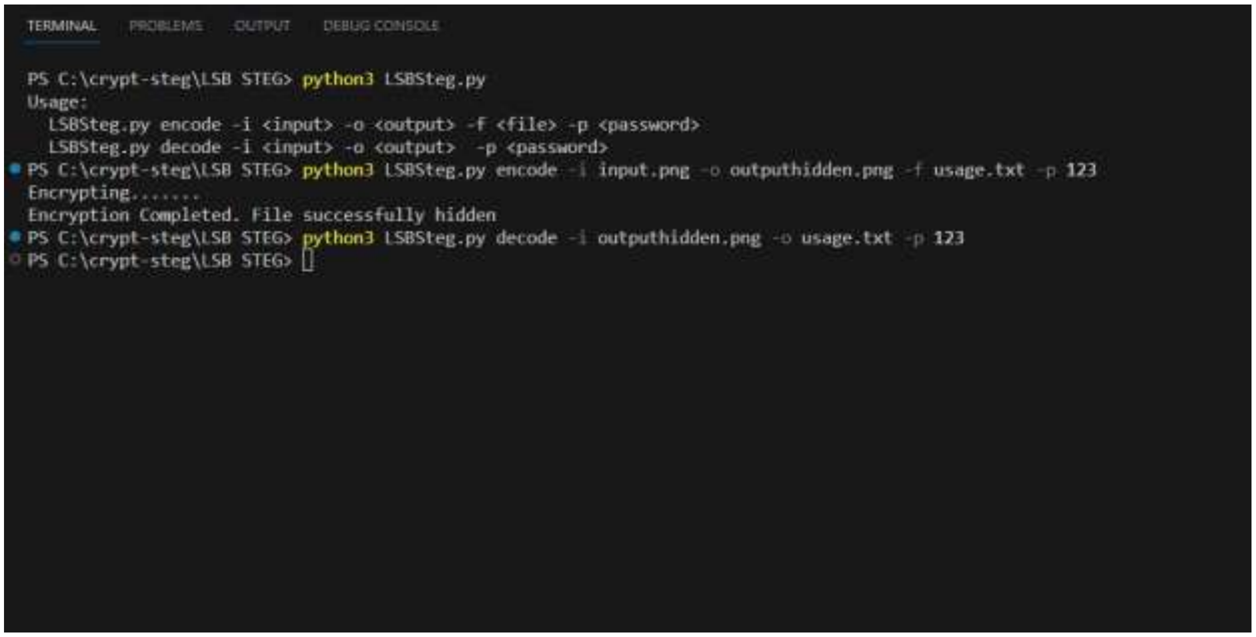
```
def GetKey(password):
    hasher = SHA256.new(password.encode('utf-8'))
    return hasher.digest()

def Test():
```

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE

```
PS C:\crypt-steg\AES encryption> & C:/Python39/python.exe "c:/crypt-steg/AES encryption/simp_AES.py"
Press 1 for Encrypt and 2 for Decrypt: 1
Type in File that you want to encrypt: usage.txt
Choose a Password: 123
Encrypting.......
Completed
PS C:\crypt-steg\AES encryption> & C:/Python39/python.exe "c:/crypt-steg/AES encryption/simp_AES.py"
Press 1 for Encrypt and 2 for Decrypt: 2
Type in Filename that you want to decrypt: usage.txt.hidn
Type in the Password: 123
Decrypting.......
Completed
PS C:\crypt-steg\AES encryption> 
```

Figure 10: Successfull ecnryption/decryption

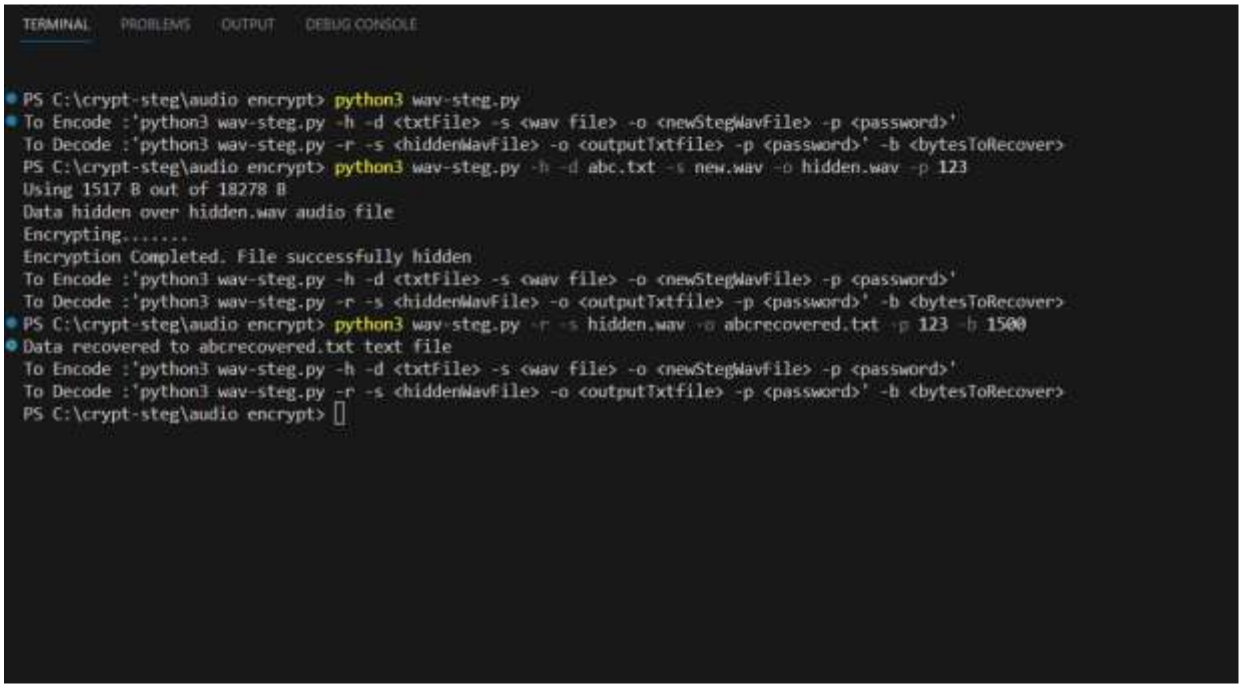
Appendix B: Encryption/Decryption of PNG file



```
PS C:\crypt-steg\LSB STEG> python3 LSBSteg.py
Usage:
  LSBSteg.py encode -i <input> -o <output> -f <file> -p <password>
  LSBSteg.py decode -i <input> -o <output>  -p <password>
PS C:\crypt-steg\LSB STEG> python3 LSBSteg.py encode -i input.png -o outputhidden.png -f usage.txt -p 123
Encrypting.......
Encryption Completed. File successfully hidden
PS C:\crypt-steg\LSB STEG> python3 LSBSteg.py decode -i outputhidden.png -o usage.txt -p 123
PS C:\crypt-steg\LSB STEG> []
```

Figure 11: Successful encryption and decryption to PNG file

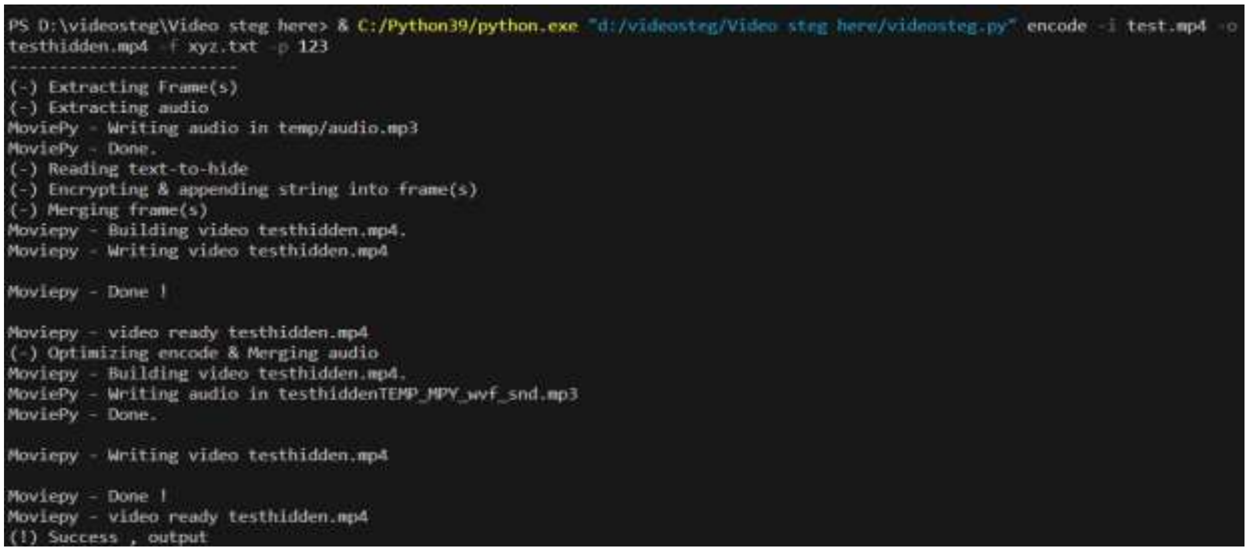
Appendix C: Audio Steganography



```
PS C:\crypt-steg\audio encrypt> python3 wav-steg.py
To Encode :'python3 wav-steg.py -h -d <txtFile> -s <wav file> -o <newStegWavFile> -p <password>'
To Decode :'python3 wav-steg.py -r -s <hiddenWavFile> -o <outputTxtfile> -p <password>' -b <bytesToRecover>
PS C:\crypt-steg\audio encrypt> python3 wav-steg.py -h -d abc.txt -s new.wav -o hidden.wav -p 123
Using 1517 B out of 18278 B
Data hidden over hidden.wav audio file
Encrypting.......
Encryption Completed. File successfully hidden
To Encode :'python3 wav-steg.py -h -d <txtFile> -s <wav file> -o <newStegWavFile> -p <password>'
To Decode :'python3 wav-steg.py -r -s <hiddenWavFile> -o <outputTxtfile> -p <password>' -b <bytesToRecover>
PS C:\crypt-steg\audio encrypt> python3 wav-steg.py -r -s hidden.wav -o abcrecovered.txt -p 123 -b 1500
Data recovered to abcrecovered.txt text file
To Encode :'python3 wav-steg.py -h -d <txtFile> -s <wav file> -o <newStegWavFile> -p <password>'
To Decode :'python3 wav-steg.py -r -s <hiddenWavFile> -o <outputTxtfile> -p <password>' -b <bytesToRecover>
PS C:\crypt-steg\audio encrypt> []
```

Figure 12: Successful Audio Stegenograph

Appendix D: Video Steganography



```
PS D:\videosteg\Video steg here> & C:/Python39/python.exe "d:/videosteg/Video steg here/videosteg.py" encode -i test.mp4 -o
testhidden.mp4 -f xyz.txt -p 123
------------------------
(-) Extracting Frame(s)
(-) Extracting audio
MoviePy - Writing audio in temp/audio.mp3
MoviePy - Done.
(-) Reading text-to-hide
(-) Encrypting & appending string into frame(s)
(-) Merging frame(s)
Moviepy - Building video testhidden.mp4.
Moviepy - Writing video testhidden.mp4

Moviepy - Done !

Moviepy - video ready testhidden.mp4
(-) Optimizing encode & Merging audio
Moviepy - Building video testhidden.mp4.
MoviePy - Writing audio in testhiddenTEMP_MPY_wvf_snd.mp3
MoviePy - Done.

Moviepy - Writing video testhidden.mp4

Moviepy - Done !
Moviepy - video ready testhidden.mp4
(!) Success , output
```

Figure 13: Extract-encrypt-merge

Appendix E:Original/Stegano Image

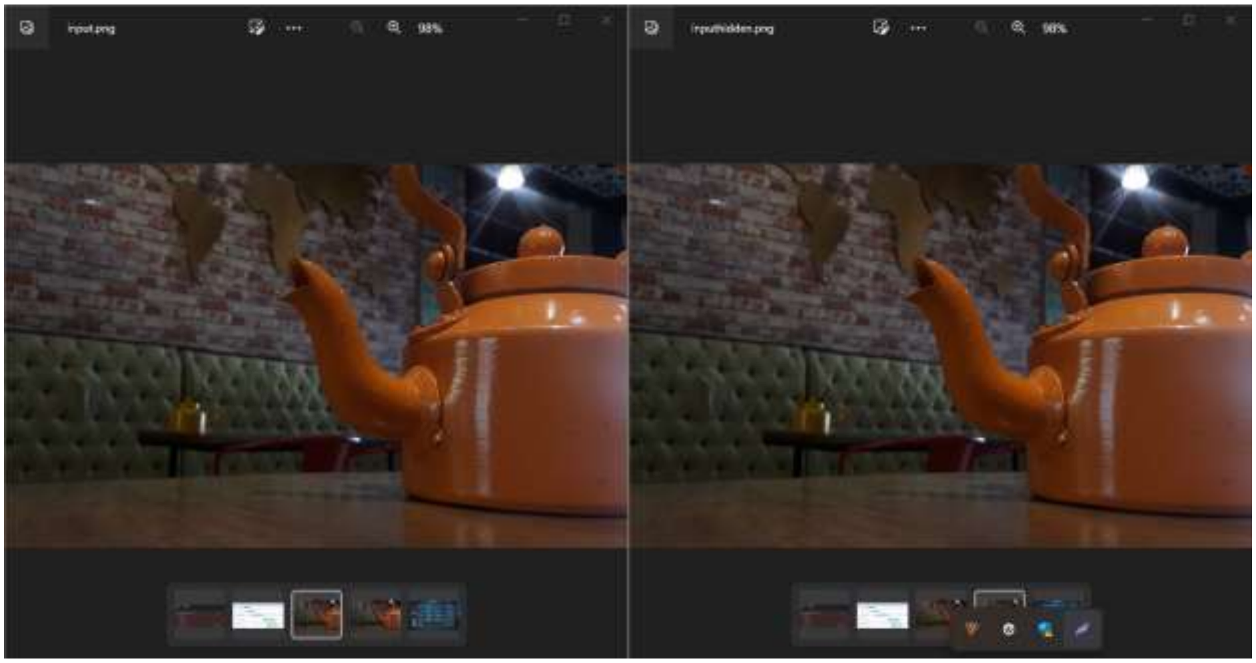


Figure 14: Comparign Original and Hidden file

BIBLIOGRAPHY

[1] Incremental Model, S. Pressman, Software Engineering Fundamentals, Seventh Edition.[offline]

[2] Mike Driscoll: An Intro to encryption in Python3 [https://dzone.com/articles/an-intro-to-encryption-in-python-3[offline]

[3] Image steganography [https://www.dreamincode.net/forums/topic/38678-image-steganography/[14 Aug,2018]

[4] LSB-DCT based Image steganography[https://stackoverflow.com/questions/35396977/lsb-dct-based-image steganography][14 Aug,2018]

[5] Kaur, Harmeet, and Amandeep Kaur. "A Survey of Steganography Techniques." International Journal of Advanced Research in Computer Science, vol. 9, no. 2, 2018[offline].

[6] Cryptography and Steganography with Python, [https://opensourceforu.com/2010/05/cryptography-and-steganography-with-python/][16Aug2018]

[7] An Overview of Steganography James Madison University InfoSec Tech report Department of Computer Science [http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.137.5129&rep=rep1&type=pdf][20 Aug2018]

[8] Stallings, William. "Cryptography and Network Security: Principles and Practice." Pearson, 2017.

[9] Best available Tools to hide data: [https://www.geekdashboard.com/best-steganography-tools/#openstego][21 Aug 2018]

[10] Dr. Joyti Preet: Image steganography [https://www.slideshare.net/hussainsavani/image-steganography][22 Aug 2018]

[11] Jayaram p,Rangaanatha H R, Anupama H S: INFORMATION HIDING USING AUDIO STEGANOGRAPHY–A SURVEY[http://aircconline.com/ijma/V3N3/3311ijma08.pdf][22 Aug 2018]

[12] Jain, Rohit, and Manpreet Singh. "A Comparative Analysis of Cryptography Techniques." International Journal of Advanced Computer Research, vol. 3, no. 2, 2013

[13] Key Distribution for Symmetric Key Cryptography: A Reviewby Yashaswini J [http://www.rroij.com/open-access/key-distribution-for-symmetric-keycryptography-a-review.php?aid=56128][23 Aug 2018]

[14] AES-256 with random key generation instead of hash: [https://crypto.stackexchange.com/questions/50359/aes-256-with-random-key-generation-instead-of-hash][10sept 2018]

[15] Tzeng, Wei-Kuang, and Chaoping Li. "Cryptography: An Introduction." Journal of Applied Sciences, vol. 11, no. 11, 2011.

[16] Wayner, Peter. "Disappearing Cryptography: Information Hiding: Steganography & Watermarking." Morgan Kaufmann, 2002.

[17] Python GUI – tkinter,[https://www.geeksforgeeks.org/python-gui-tkinter/][22 Aug2018]

[18] Tutorial for cryptography [http://play.google.com/store/apps/details?id= com.gd.tutorialforcryptography][12sept 2018]

[19] Menezes, Alfred J., Jonathan Katz, and Paul C. van Oorschot. "Handbook of Applied Cryptography." CRC Press, 1996.