Encryption System

with image steganography

King Saud University

College of Computer and Information Sciences

Cybersecurity Joint Master's Program

SEC 503 Applied Cryptography

Course Project

Prepared by

Amani Alghwainm

Ghadah Alkhodhiry

Supervised by

Dr. Abdulrahman AlMutairi

ABSTRACT

Data hiding is the art of hiding data for various purposes such as, to maintain private data, secure confidential data and so on. Securely exchange the data over the internet network is very important issue. So, to transfer the data securely to the many approaches like cryptography and steganography. In this destination, there are project we propose a small app that can encrypt and decrypt text and files with AES encryption, as well as hide encrypted text data inside images.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 : INTRODUCTION

1.1 INTRODUCTION

The growing use of Internet needs to take attention while we send and receive personal information in a secured manner. For this, there are many approaches that can transfer the data into different forms so that their resultant data can be understood if it can be returned into its original form. This technique is known as encryption. However, a major disadvantage of this method is that the existence of data is not hidden. If someone gives enough time, then the unreadable encrypted data may be converted into its original form.

A solution to this problem has already been achieved by using a “steganography” technique to hide data in a cover media so that other cannot notice it. The characteristics of the cover media depends on the amount of data that can be hidden, the perceptibility of the message and its robustness.

1.2 STATEMENT OF PROBLEM

This project addresses the security problem of transmitting the data over internet network, the main idea coming when we start asking that how can we send a message secretly to the destination? The science of steganography answers this question. Using steganography, information can be hidden in carriers such as images, audio files, text files, videos, and data transmissions.

1.3 PROJECT OBJECTIVES

The main objectives of our project are to product security tool based on steganography techniques to hider message carried by stego-media which should not be sensible to human beings and avoid drawing suspicion to the existence of hidden message. Combine steganography with cryptography.

1.4 ARCHITECTURE & COMPONENTS

This system provides the user encrypt and decrypt the text with random key by use AES 128, in encryption the secret information is hiding in with image file, and on the other side the decryption is getting the hidden information from the stego image file.

1.5 BACKGROUND INFORMATION

1.5.1 STEGANOGRAPHY

Steganography is the art of hiding a secret message inside of (or even on top of) something that is not secret. the word Steganography literally means covered or hiding writing as derived from Greek. Steganography has its place in security. It is not intended to replace cryptography but supplement it. [1]

Hiding a message with Steganography methods reduces the chance of a message being detected. If the message is also encrypted, then it provides another layer of protection. Therefore, some Steganographic methods combine traditional Cryptography with Steganography; the sender encrypts the secret message prior to the overall communication process, as it is more difficult for an attacker to detect embedded cipher text in a cover. It has been used through the ages by ordinary people, spies, rulers, government, and armies. There are many stories about Steganography. [1]

For example, ancient Greece used methods for hiding messages such as hiding In the field of Steganography, some terminology has developed. The adjectives 'cover', 'embedded', and 'stego' were defined at the information hiding workshop held in Cambridge, England. The term "cover" refers to description of the original, innocent massage, data, audio, video, and so on. Steganography is not a new science; it dates back to ancient times. [1]

Hidden information in the cover data is known as the "embedded" data and information hiding is a general term encompassing many sub disciplines, is a term around a wide range of problems beyond that of embedding message in content.

The term hiding here can refer to either making the information undetectable or keeping the existence of the information secret. [1]

Information hiding is a technique of hiding secret using redundant cover data such as images, audios, movies, documents, etc. This technique has recently become important in a number of application areas. For example, digital video, audio, and images are increasingly embedded with imperceptible marks, which may contain hidden signatures or watermarks that help to prevent unauthorized copy. It is a performance that inserts secret messages into a cover file, so that the existence of the messages is not apparent. [1]

Research in information hiding has tremendous increased during the past decade with commercial interests driving the field. Although the art of concealment “hidden information” as old as the history, but the emergence of computer and the evolution of sciences and techniques breathe life again in this art with the use of new ideas, techniques, drawing on the computer characteristics in the way representation of the data, well-known computer representation of all data including (Multimedia) is binary these representations are often the digital levels and areas and change values-aware of slight not aware or felt by Means sensual of human such as hearing, sight, the advantage use of these properties to hide data in multimedia by replace the values of these sites to the values of data to be hidden, taking into account the acceptable limits for the changeover, and not exceeded to prevent degradation media container with a change becomes aware and felt by human. It should be noted here that although the art of hidden information come in the beginning of the computer and its techniques However, the seriousness of the work in the stenography as a stand-alone science started in 1995. [1]

1.5.2 AES ALGORITHM

The AES algorithm (also known as the Rijndael algorithm) is a symmetrical block cipher algorithm that takes plain text in blocks of 128 bits and converts them to ciphertext using keys of 128, 192, and 256 bits. Since the AES algorithm is considered secure, it is in the worldwide standard. [3]

The AES algorithm uses a substitution-permutation, or SP network, with multiple rounds to produce ciphertext. The number of rounds depends on the key size being used. A 128-bit key size dictates ten rounds, a 192-bit key size dictates 12 rounds, and a 256-bit key size has 14 rounds. Each of these rounds requires a round key, but since only one key is inputted into the algorithm, this key needs to be expanded to get keys for each round, including round 0. [3]

Each round in the algorithm consists of four step:

- Substitution of the bytes
  In the first step, the bytes of the block text are substituted based on rules dictated by predefined S-boxes (short for substitution boxes).
- Shifting the rows
  Next comes the permutation step. In this step, all rows except the first are shifted by one.
- Mixing the columns
  In the third step, the Hill cipher is used to jumble up the message more by mixing the block's columns.
- Adding the round key
  In the final step, the message is XORed with the respective round key.

CHAPTER 2 : SYSTEM ANALYSIS

2.1 USE CASE DESCRIPTION

A use case is a written description of how users will perform tasks on the system. It outlines, from a user's point of view, a system's behavior as it responds to a request. Each use case is represented as a sequence of simple steps, beginning with a user's goal, and ending when that goal is fulfilled. [4]

The use case description of our project as follows.

Actors	User
Description	The user wants encrypt text message and hide it in image.
Data	Cover image, Message
Response	1.Click on generate key button. 2.Enter the secret message in the textbox. 3.Click on Encrypt button. 4.The encrypted message will be saved and shown in the next textbox. 5.Click on browser button. 6.Select image. 7.Click on hide button. 8.The message will be embedded into the selected image. 9.Stego-image file will be saved.

Table 2-1: Use case description – Encryption & Embedding process

Actors	User
Description	The user wants to extract message from Stego-image file.
Data	Stego-Image
Response	1.Enter the generated public key. 2.Click on browser button. 3.Select the Stego-Image file. 4.Click on Show button. 5.The encrypted message will be shown in the next textbox. 6.Click on Decrypt button. 7.The secret message will be shown in the first textbox.

Table 2-2: Use case description – Extracting & Decryption process

2.2 Use Case Diagram

A use case diagram can summarize the details of the system's users (actors) and their interactions with the system. An actor in the UML "specifies a role played by a user or any other system that interacts with the subject." [5]

There are two actors in our application:

1.User

2. Program

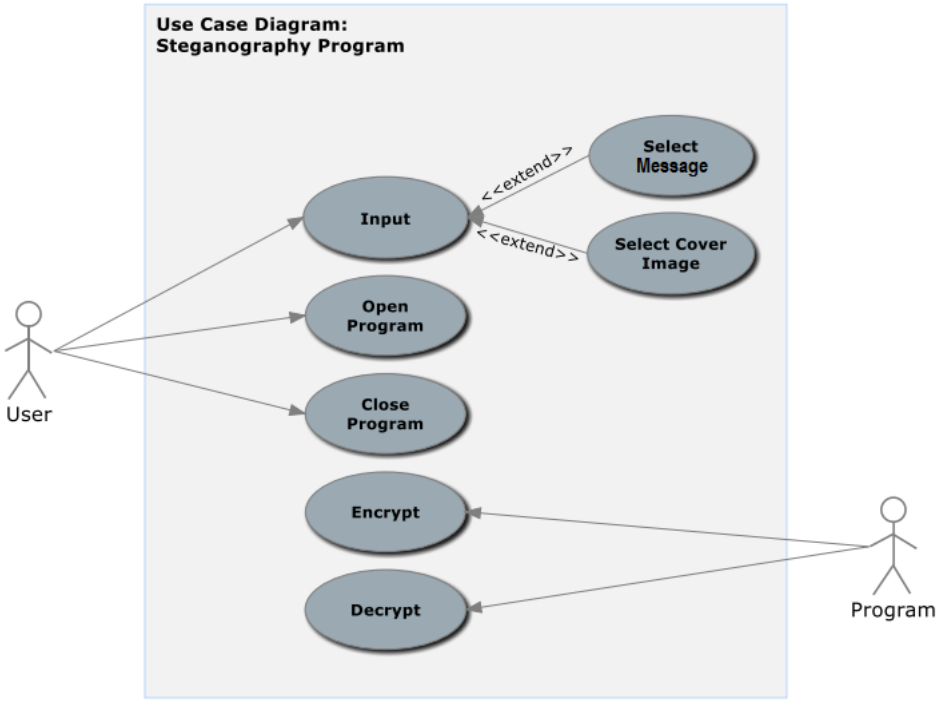

Figure 2-1: Use case diagram

2.3 Hardware and Software Requirements

The execution phase was developed based upon two phases. Encryption & Embedding and Extracting & Decryption phases. We require few hardware and software interfaces for implementing these two phases.

The software interface which is implemented in this project is done using Python 3.9 or higher.

CHAPTER 3 : SYSTEM DESIGN

3.1 Class Diagram

A class diagram is a picture for describing generic descriptions of possible systems. Class diagrams and collaboration diagrams are alternate representations of object models. Class diagrams contain classes and object diagrams contain objects, but it is possible to mix classes and objects when dealing with various kinds of metadata, so the separation is not rigid. [9]

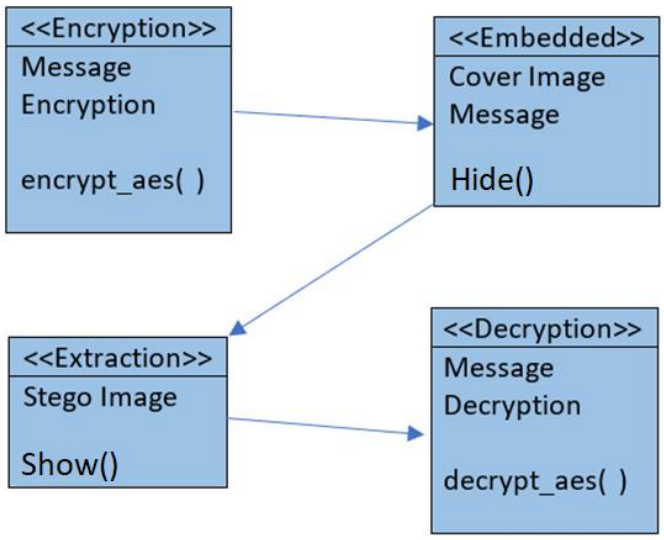

Figure 3-1: Class diagram

3.2 Sequence Diagram

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. [10]

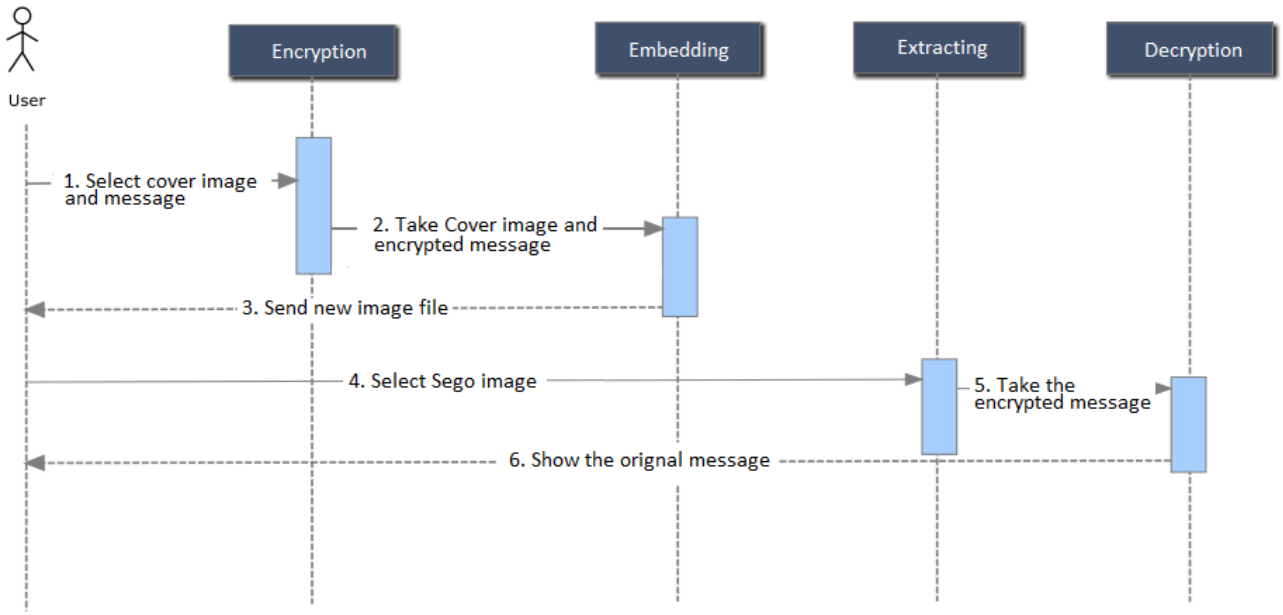

Figure 3-2: Sequence diagram

3.3 Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration, and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by step workflows of components in a system. An activity diagram shows the overall flow of processes. [11]

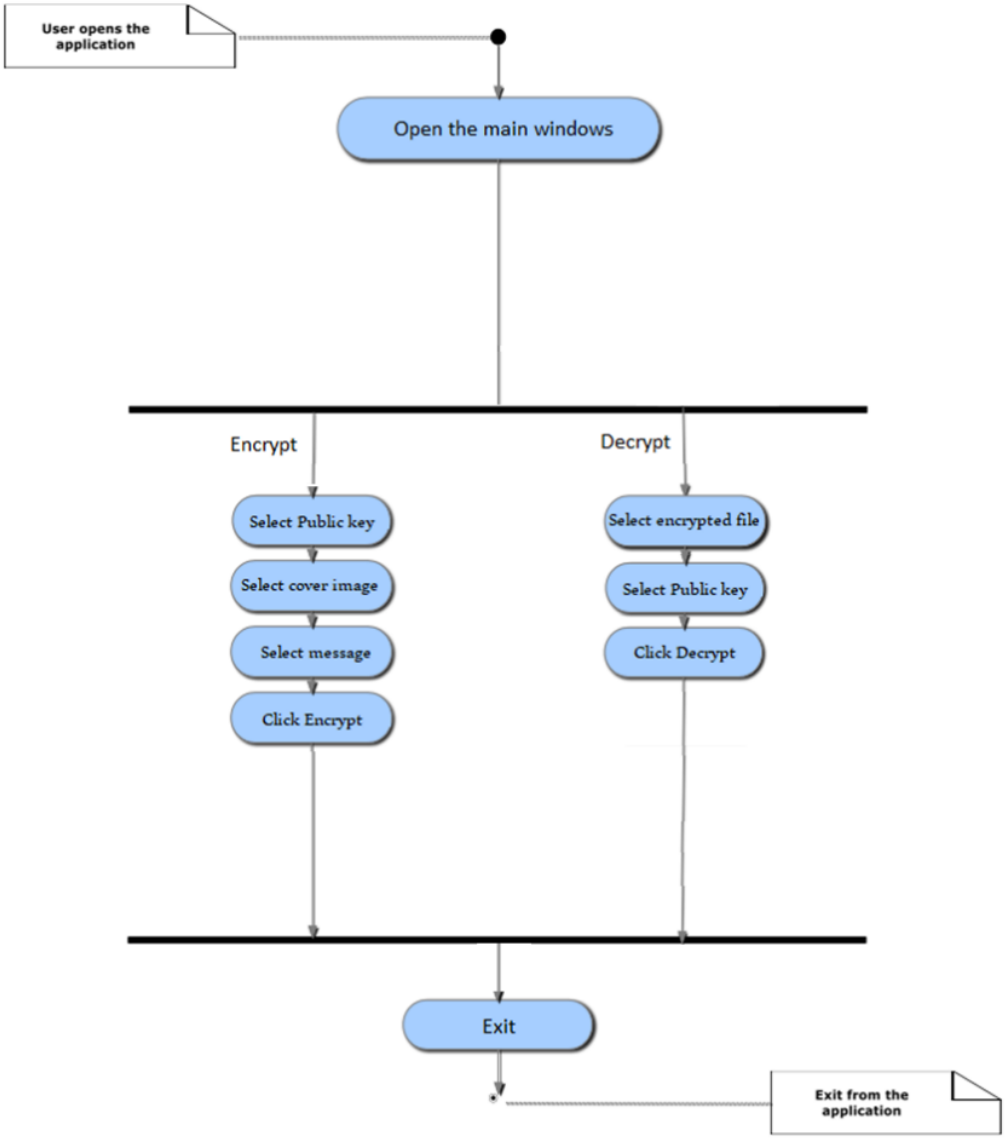

Figure 3-3: System Activities Activity Diagram

CHAPTER 4 : IMPLEMENTATION

SCREEN SHOTS OF THE SYSTEM

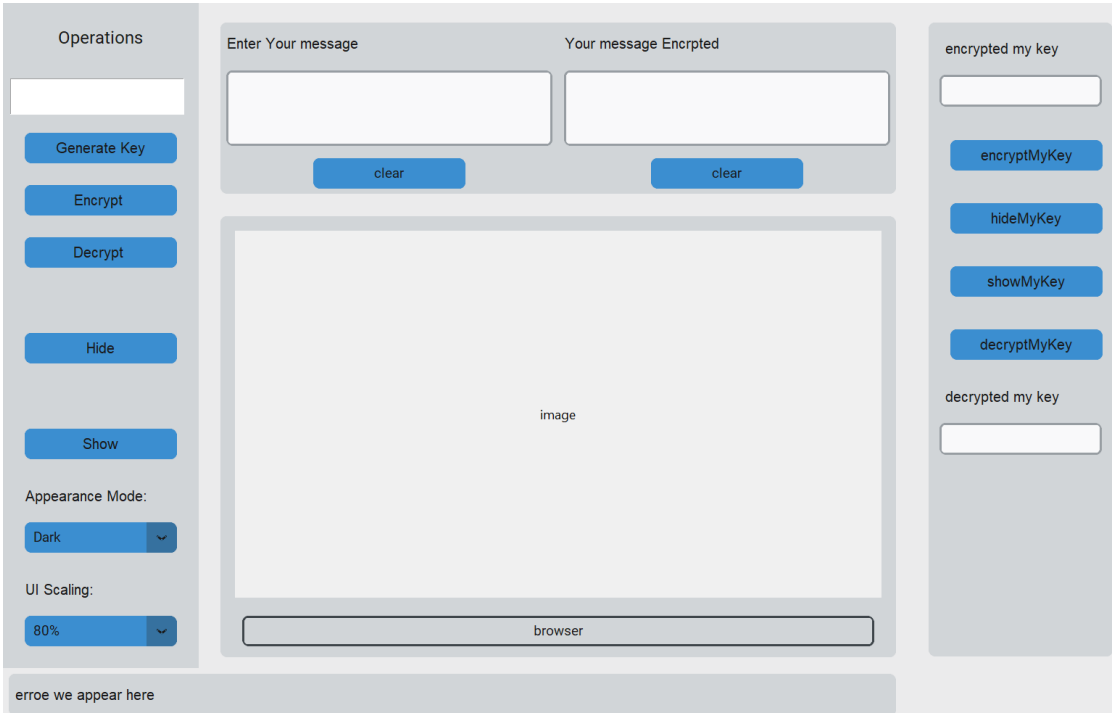

Figure 4-1: Main Screen

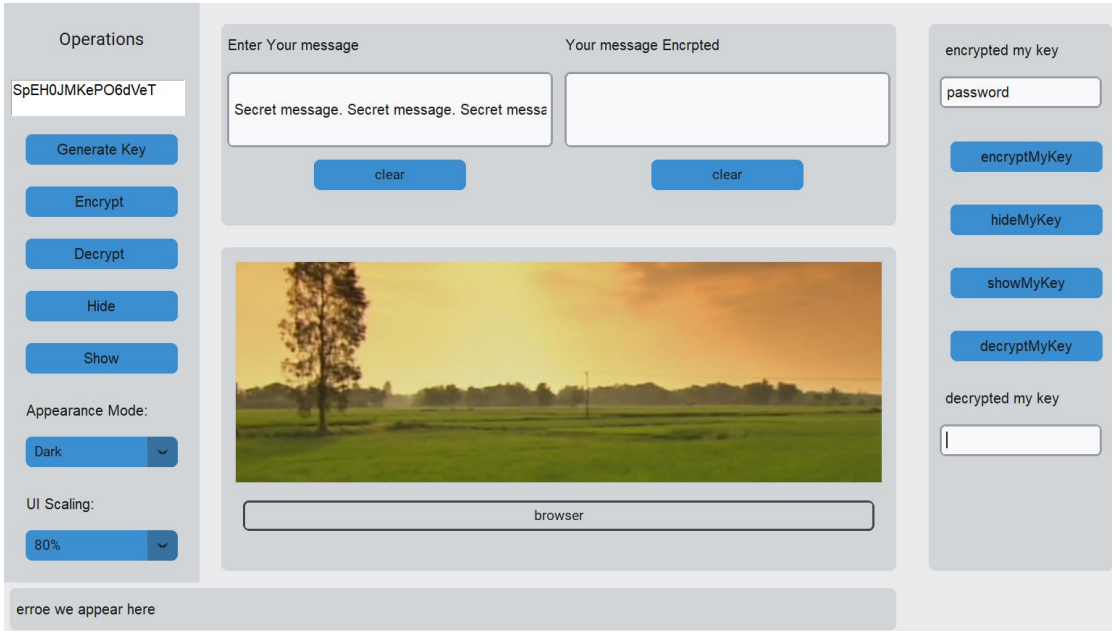

Figure 4-2: Encryption & Embedding Process

To Start Encryption & Embedding Process

1.First, click on generate key button.

2.The message's encryption key will be generated.

2.Enter your secret message in the textbox.

3.Click on Encrypt button.

4.The encrypted message will be saved and shown in the next textbox.

5.Click on browser button.

6.Select image.

7.Click on hide button.

8.The encrypted message will be embedded into the selected image.

9.Stego-image file will be saved.

To Encrypt & Embedding the secret random key with the message

1.Enter the key's encryption key.

2.Click on encryptMyKey button.

3.The encrypted key will be shown below.

4.Click on browser button.

6.Select the Stego-image.

7.Click on hideMyKey button.

8.The encrypted key will be embedded into the selected image.

9.New Stego-image file will be saved.

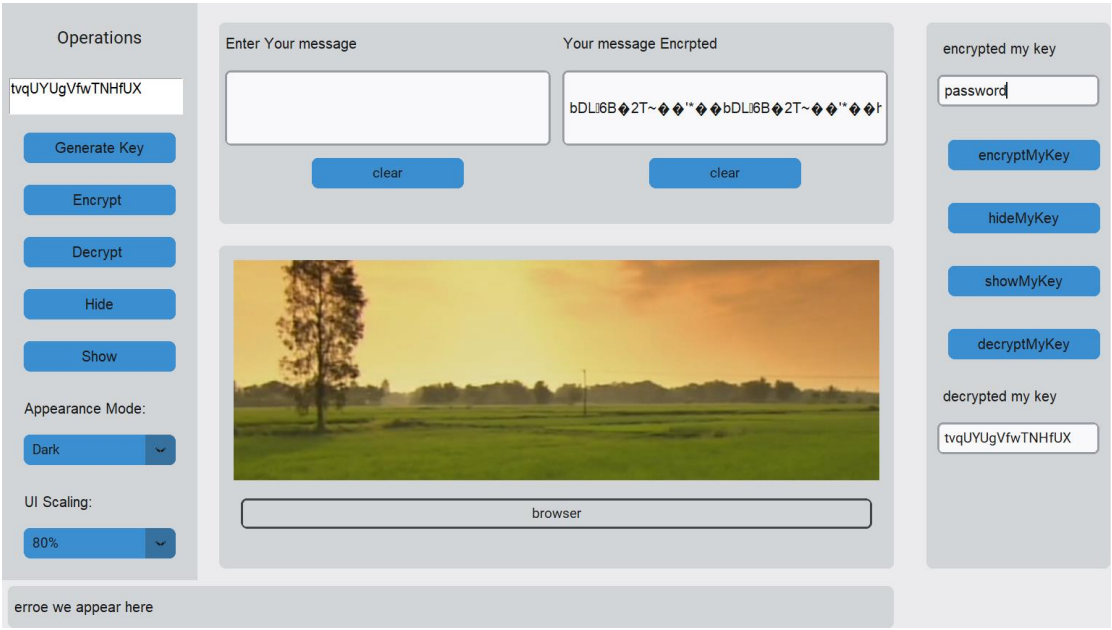

Figure 4-3: Extracting & Decryption Process

<u>To Start Extracting & Decryption Process</u>

1.Enter the message's encryption key.

2.Click on browser button.

3.Select the Stego-Image file.

4.Click on Show button.

5.The encrypted message will be shown in the textbox.

6.Click on Decrypt button.

7.The secret message will be shown in the first textbox.

<u>To Extracting & Decrypt the encrypted key</u>

1.Enter the key's encryption key.

2.Select the Stego-image.

3.Click on ShowMyKey.

4.The encrypted key will be shown below.

5.Click on decryptMyKey button.

6.The message's encryption key will be shown.

CHAPTER 5 : CONCLUSION AND FUTURE RECOMMENDATION

5.1 CONCLUSIONS

The meaning of Steganography is hiding information and the related technologies. There is a principal difference between Steganography and Encryption; however they can meet at some points too. They can be applied together, i.e. encrypted information can be hidden in addition. To hide something a covering medium is always needed (Picture, sound track, text or even the structure of a file system, etc.). The technology of hiding should match the nature of the medium. The hidden information should not be lost, if the carrying medium is edited, modified, formatted, re-sized, compressed or printed. Nowadays the most popular application of Steganography is hiding copy rights and other commercial information. Such kind of hidden information is known as e-watermark. The e-watermark is not always invisible. There are cases when it is made deliberately strikingly visible e.g. in case of trial versions of software.

5.2 FUTURE RECOMMENDATIONS

The major limitation of the application is designed for images cover files. It accepts only images as a carrier file. The future work on this project is to improve the compression ratio of the image to the text. This project can be extended to a level such that it can be used for the different types of multimedia files.

The future work on this project is to extend to a level such that it can be used for the different types of multimedia files.

REFERENCES

[1] http://www.iaeng.org/publication/WCECS2010/WCECS2010_pp144-148.pdf

[2] https://www.geeksforgeeks.org/rsa-algorithm-cryptography/

[3] https://www.techtarget.com/searchsecurity/definition/Advanced-Encryption-Standard

[4] https://www.pearsonhighered.com/assets/samplechapter/0/2/0/1/0201709139.pdf

[5] http://csis.pace.edu/~marchese/CS389/L9/Use%20Case%20Diagrams.pdf

[6] https://pypi.org/project/PySimpleGUI/

[7] https://pypi.org/project/pycryptodome/

[8] https://pypi.org/project/Pillow/

[9] https://repository.dinus.ac.id/docs/ajar/Pertemuan_6_-_UML_Class_Diagrams.pdf

[10] http://csis.pace.edu/~marchese/CS389/L9/Sequence%20Diagram%20Tutorial.pdf

[11] https://www.kdkce.edu.in/pdf/Unit-5%20Activity%20Diagrams-1.pdf

APPENDICES

encryption_algorithm.py file

```python
import aes128
import os

class encryption_algorithm:
    def encrypt(self, d, key):
        with open(d, "rb") as f:
            data = f.read()

        crypted_data = []
        temp = []
        for byte in data:
            temp.append(byte)
            if len(temp) == 16:
                crypted_part = aes128.encrypt(temp, key)
                crypted_data.extend(crypted_part)
                del temp[:]
        else:
            # padding v1
            # crypted_data.extend(temp)

            # padding v2
            if 0 < len(temp) < 16:
                empty_spaces = 16 - len(temp)
                for i in range(empty_spaces - 1):
                    temp.append(0)
                temp.append(1)
                crypted_part = aes128.encrypt(temp, key)
                crypted_data.extend(crypted_part)

            # return (bytes(crypted_data))

            out_path = os.path.join(
                os.path.dirname(d), "crypted_" + os.path.basename(d)
            )
            if os.path.exists(out_path):
                os.remove(out_path)
            with open(out_path, "xb") as ff:
                ff.write(bytes(crypted_data))
```

encryption_algorithm.py file

```
39          #  return bytes(crypted_data).decode("utf-8")
40
41      def decypher(self, d, options, key):
42          if options == 1:
43              data = bytes(d)
44          if options == 2:
45              with open(d, "rb") as f:
46                  data = f.read()
47
48
49          decrypted_data = []
50          temp = []
51          for byte in data:
52              temp.append(byte)
53              if len(temp) == 16:
54                  decrypted_part = aes128.decrypt(temp
    , key)
55                  decrypted_data.extend(decrypted_part)
56                  del temp[:]
57          else:
58              # padding v1
59              # decrypted_data.extend(temp)
60
61              # padding v2
62              if 0 < len(temp) < 16:
63                  empty_spaces = 16 - len(temp)
64                  for i in range(empty_spaces - 1):
65                      temp.append(0)
66                  temp.append(1)
67                  decrypted_part = aes128.encrypt(temp
    , key)
68                  decrypted_data.extend(decrypted_part)
69
70              if options == 1:
71                  return bytes(decrypted_data).decode(
    errors="replace")
72              # return bytes(decrypted_data).decode("
    utf-8")
73              out_path = os.path.join(
74                  os.path.dirname(d), "decrypted_" + os
    .path.basename(d)
```

encryption_algorithm.py file

```
            )
            if os.path.exists(out_path):
                os.remove(out_path)
            with open(out_path, "xb") as ff:
                ff.write(bytes(decrypted_data))

```

hide.py file

```python
from PIL import Image

def en_hide(enc_img,option=1):
    # Creating an Image Object
    enc_img = Image.open(enc_img)
    # Loading pixel values of original image, each entry is pixel value ie., RGB values as sublist
    enc_pixelMap = enc_img.load()
    # Creating an empty String for our hidden message
    msg = ""
    msg_index = 0

    if option ==2:
        # Traversing through the pixel values
        for row in range(enc_img.size[0]):
            for col in range(enc_img.size[1]):
                # Fetching RGB value a pixel to sublist
                list = enc_pixelMap[row, col]
                g = list[1]
                if col == 0 and row == 0:
                    msg_len = g
                elif msg_len > msg_index:
                    # Reading the message from R value of pixel
                    msg = msg + chr(g)  # Converting to charac
                    msg_index = msg_index + 1
    else:
        # Traversing through the pixel values
        for row in range(enc_img.size[0]):
            for col in range(enc_img.size[1]):
                # Fetching RGB value a pixel to sublist
                list = enc_pixelMap[row, col]
                r = list[0]
                if col == 0 and row == 0:
                    msg_len = r
                elif msg_len > msg_index:
                    # Reading the message from R value of pixel
```

hide.py file

```
36                      msg = msg + chr(r)  # Converting
   to charac
37                      msg_index = msg_index + 1
38      enc_img.close()
39      return msg
40
41
42  def hide_image(path, msg,option=1):
43
44      # Creating a image object
45      org_img = Image.open(path)
46      # Loading pixel values of original image, each
   entry is pixel value ie., RGB values as sublist
47      org_pixelMap = org_img.load()
48
49      # Creating new image object with image mode and
   dimensions as that of original image
50      enc_img = Image.new(org_img.mode, org_img.size)
51      enc_pixelsMap = enc_img.load()
52
53      # Reading message to be encrypted from user
54      msg_index = 0
55      # Finding the lenght of messag
56      msg_len = len(msg)
57
58      if option ==2:
59          # Traversing through the pixel values
60          for row in range(org_img.size[0]):
61              for col in range(org_img.size[1]):
62                  # Fetching RGB value a pixel to
   sublist
63                  list = org_pixelMap[row, col]
64                  r = list[0]  # R value
65                  g = list[1]  # G value
66                  b = list[2]  # B value
67                  if row == 0 and col == 0:
68                      ascii = msg_len
69                      enc_pixelsMap[row, col] = (r,
   ascii, b)
70                  elif msg_index <= msg_len:
71                      # Hiding our message inside the R
```

hide.py file

```
 values of the pixel
                    c = msg[msg_index - 1]
                    ascii = ord(c)
                    enc_pixelsMap[row, col] = (r,
ascii,b)
                else:  # Assigning the pixel values
of old image to new image
                    enc_pixelsMap[row, col] = (r, g
, b)
                msg_index += 1

    else:
        # Traversing through the pixel values
        for row in range(org_img.size[0]):
            for col in range(org_img.size[1]):
                # Fetching RGB value a pixel to
sublist
                list = org_pixelMap[row, col]
                r = list[0]  # R value
                g = list[1]  # G value
                b = list[2]  # B value
                if row == 0 and col == 0:
                    ascii = msg_len
                    enc_pixelsMap[row, col] = (ascii
, g, b)
                elif msg_index <= msg_len:
                    # Hiding our message inside the
R values of the pixel
                    c = msg[msg_index - 1]
                    ascii = ord(c)
                    enc_pixelsMap[row, col] = (ascii
, g, b)
                else:  # Assigning the pixel values
of old image to new image
                    enc_pixelsMap[row, col] = (r, g
, b)
                msg_index += 1

    org_img.close()
    # Display the image
    enc_img.show()
```

hide.py file

```
    # Save the image
    enc_img.save("encrypted_image.png")
    enc_img.close()

```