Project 3: Steganography

Name: Shefali Athavale

Approach:

For this project, I wrote a Steganography Program using Python. The program is written for JPEG image files only. I used the following libraries for the same - os, hashlib, binascii, re. The program gives the user a choice among the two - Embed or Extract. If the user selects 'embed'(i.e. choice '1'), the program executes the 'embed()' function. If the user selects extract(i.e. choice '2'), the program executes the extract()' function. We will now look at both the functions individually.

embed() function:

This function is used to embed a secret message in a carrier file. If the user selects this option, the program will ask the user to input the carrier file name. It is assumed that the carrier file is present in the same folder as this script/program. If not, the user has to enter the path for the file. The program then asks the user to enter the secret message he/she wants to embed in the carrier file. We then convert the text message into hexadecimal format to embed it in the image. The program then embeds the message just before the trailer signature of the JPEG EOF offset i.e. just before the 'FF D9' signature. The program also adds 4 bytes of additional data 'FF FF FF FF' before adding the message for better retrieval of the message at the 'extract' stage. The program then outputs the details of the original and modified carrier file like the file name, size and its MD5 hash. The program also outputs the embedded secret message.

extract() function:

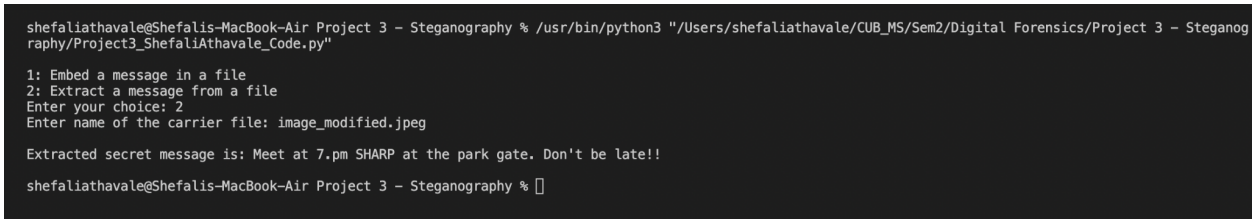
This function is used to extract the secret message from the carrier file. file. If the user selects this option, the program will ask the user to input the carrier file name(this will be the modified carrier file name). It is assumed that the carrier file is present in the same folder as this script/program. If not, the user has to enter the path for the file. The program assumes that the hidden secret message is just before the trailer signature of the JPEG EOF offset i.e. just before the 'FF D9' signature. The program then retrieves the JPEG EOF offset as well as the offset of additional bytes of data (FF FF FF FF) which we had added. The secret message is the data between 'FF FF FF FF' and 'FF D9'. The program retrieves that data, converts it from hexadecimal format to string and outputs it to the screen.

Visual Studio Code is used for writing the program. The outputs from both the files and the code is attached below.

Output:

embed():

extract():



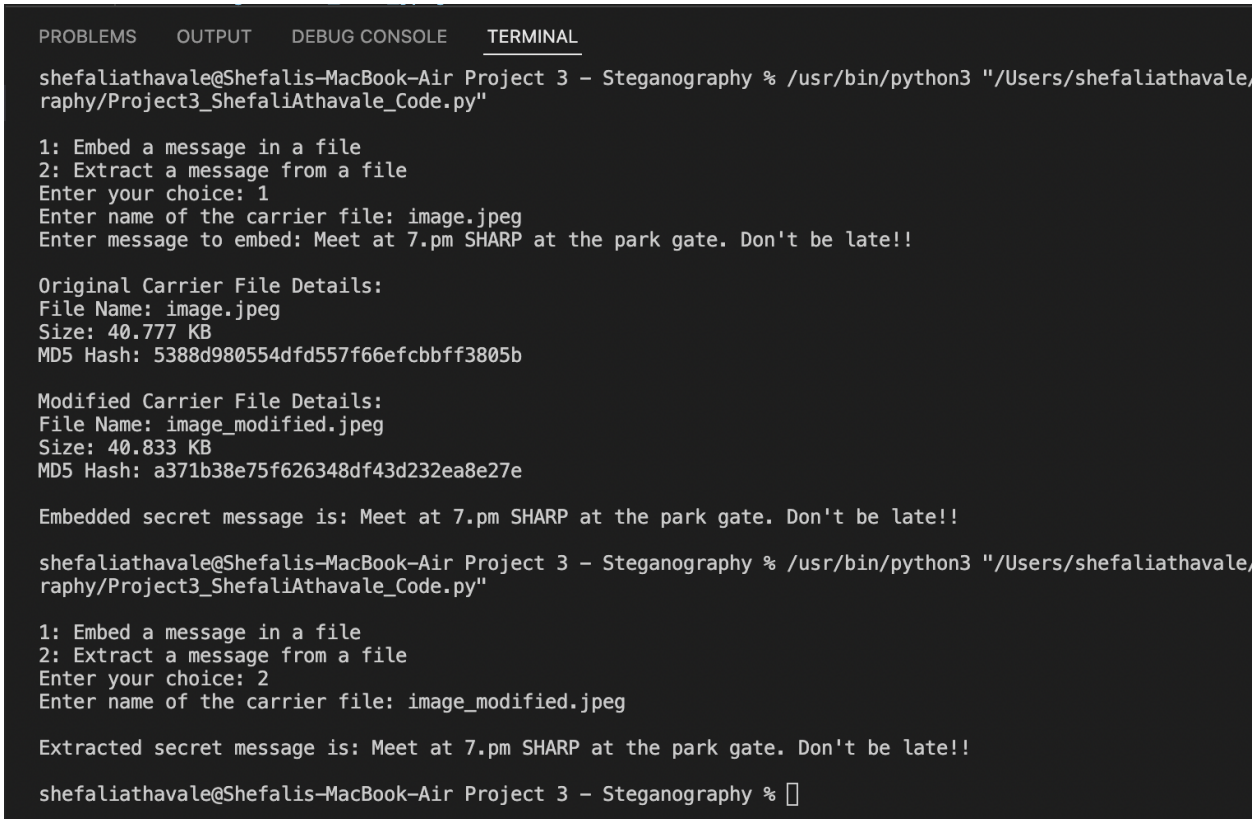
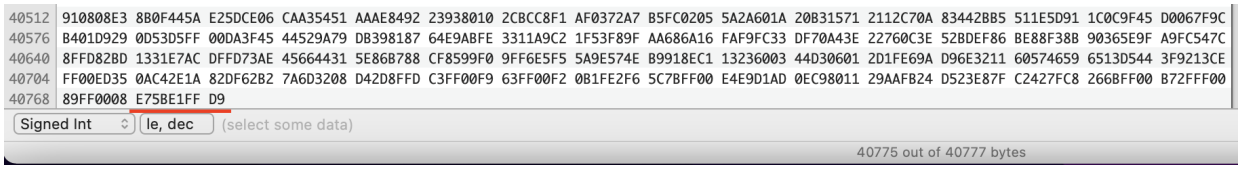
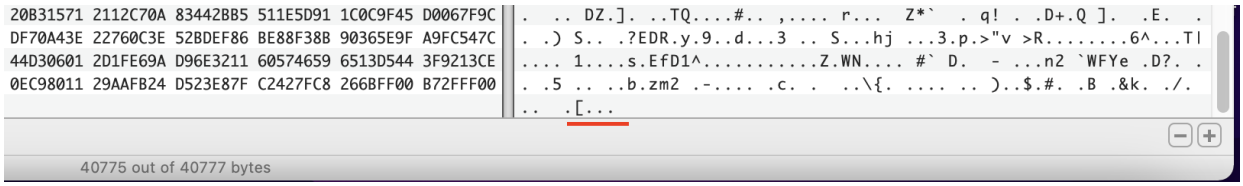
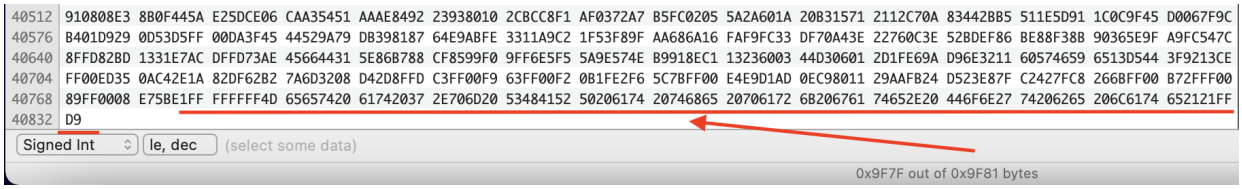


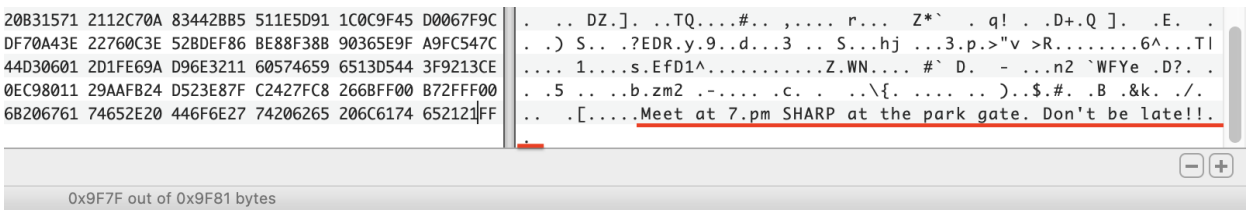
image.jpeg:





image_modified.jpeg:





Code:

```
import binascii
import re
import os
import hashlib


## Function to embed secret message in JPEG File
def embed(carrierFile,message):

   # Open carrier file and read it's contents
   with open(carrierFile, 'rb') as f:
       content = f.read()
       f.close()

   # Find the EOF offset for the JPEG file and retrieve contents of the image upto EOF
offset
   eof_list_jpeg = [match.start() for match in
re.finditer(re.escape(b'\xFF\xD9'),content)]
   eofImage = eof_list_jpeg[-1]
   hexcontent = binascii.hexlify(content[:eofImage])

   # Convert text message into bytes and add it to the original image content
   # We add 4 additional bytes of data before appending the secret message
   # for better retrieval of the secret message from the file
   msg = message.encode('utf-8')
   msg = binascii.hexlify(msg).decode('utf-8')
   dataMsg = bytes(msg, 'utf-8')
   hexcontent += b'ffffffff'+dataMsg + b'ffd9'
   data = bytes.fromhex(hexcontent.decode("utf-8"))

   # Retrieve original carrier file name
   extension = ''.join(carrierFile.split(".")[-1])
   filename = carrierFile.split(".")[:-1]

   # Write the contents to the new file
   embedFile = filename[0]+'_modified.'+extension
   with open(embedFile, 'wb') as f1:
       f1.write(data)
       f1.close()
```

```
   # Output details about Original and Modified Carrier File to the screen
   print()
   print("Original Carrier File Details:")
   print("File Name: "+carrierFile)
   print("Size: "+str(os.path.getsize(carrierFile)/1000)+" KB")
   md5Hash = hashlib.md5(content).hexdigest()
   print("MD5 Hash: "+md5Hash)
   print()
   print("Modified Carrier File Details:")
   print("File Name: "+embedFile)
   print("Size: "+str(os.path.getsize(embedFile)/1000)+" KB")
   md5Hash1 = hashlib.md5(data).hexdigest()
   print("MD5 Hash: "+md5Hash1)



## Function to extract secret message from a JPEG File
def extract(embeddedFile):

   # Open carrier file and read it's contents
   with open(embeddedFile, 'rb') as f:
       content = f.read()
       f.close()

   # Find the EOF offset for the JPEG file and retrieve contents of the image upto EOF
offset
   eof_list_jpeg = [match.start() for match in
re.finditer(re.escape(b'\xFF\xD9'),content)]
   # Find the offset of the start of 4 additional bytes of data
   eof_list_space = [match.start()+4 for match in
re.finditer(re.escape(b'\xFF\xFF\xFF\xFF'),content)]
   eofImage = eof_list_jpeg[-1]
   eofSpace = eof_list_space[-1]
   # Secret Message is the message between 4 additional bytes of data and EOF offset
for JPEG image
   secMsg = content[eofSpace:eofImage].decode('utf-8')

   return secMsg
```

```
if __name__ == "__main__":
    print()
    print("1: Embed a message in a file")
    print("2: Extract a message from a file")

    method = input("Enter your choice: ")
    if method == "1":
        filename = input("Enter name of the carrier file: ")
        message = input("Enter message to embed: ")
        embed(filename, message)
        print()
        print("Embedded secret message is: "+message)
        print()

    elif method=="2":
        filename = input("Enter name of the carrier file: ")
        secretMessage = extract(filename)
        print()
        print("Extracted secret message is: "+secretMessage)
        print()
```