Course: Image processing

Topic: Steganography and Cryptography

Program: The_secret_v2

By

Mody Morkos Kamil

Supervised by

Dr/ Marwa Refaie

Abstract:-

We live in a digital world to exchange data over the Internet, but this data may be exposed to unauthorized actions such as leakage or espionage, so we have to protect this data over the Internet as much as possible, and we can do this by merging more than one different field to ensure the strength of protection of this data. We used image processing and cryptography to encrypt and hide the secret data that we want to transfer to another party, and we will explain the techniques used in both areas to obtain security for confidential data.

Introduction:-

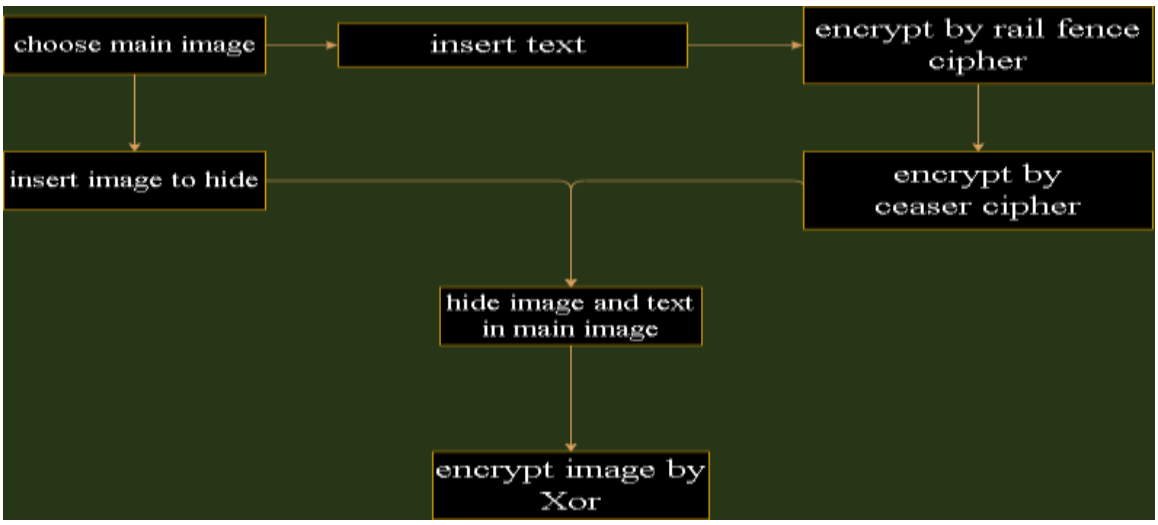
In image processing, we used Steganography, a technique through which texts or images can be hidden inside another image using two methods, the most significant bit and the least significant bit. We will explain these methods later.

And we have chosen two algorisms from cryptography, namely the Caesar cipher and the rail fence cipher, and we will explain them in them later.

In the end, we will have an encrypted image containing data in the form of encrypted text and an image

Our program workflow:-

Encryption and hiding flow



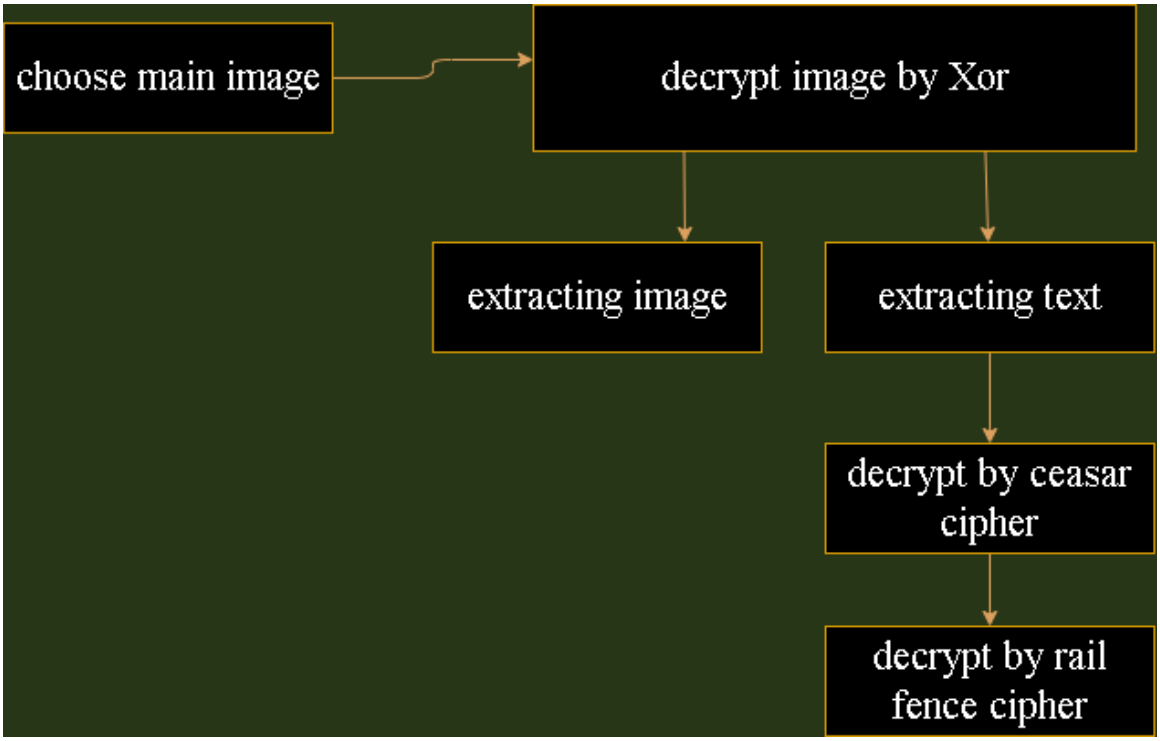
1-choose your main image to hide image and test data

2-choose your secret image that you want to hide

3-choose your secret text that you want to hide (encrypted by two algorisms)

4-save your final image that has image and test

Decryption and extracting flow:-



1-choose your main image that has secret image/text

2-extract the secret text from image (decrypted by two algorisms)

3-extract the secret image from image

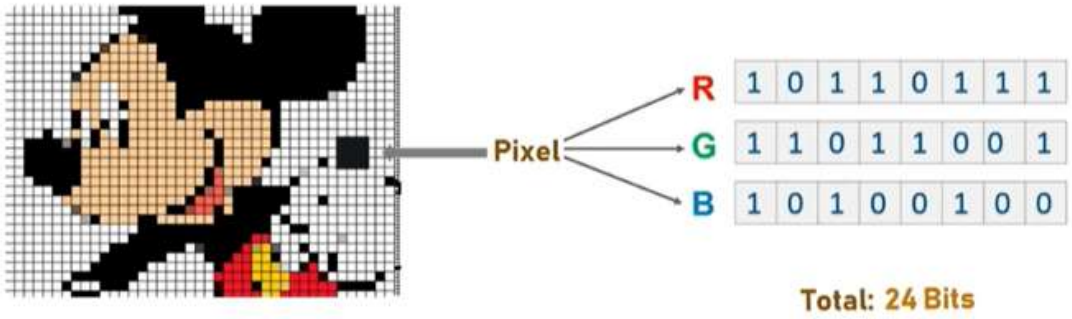
Steganography:-

Steganography is the process of hiding a secret message within a larger one in such a way that someone can not know the presence or contents of the hidden message. The purpose of Steganography is to maintain secret communication between two parties. Unlike cryptography, which conceals the contents of a secret message, steganography conceals the very fact that a message is communicated. Although steganography differs from cryptography, there are many analogies between the two, and some authors

classify steganography as a form of cryptography since hidden communication is a type of secret message.

Basic concepts of Pixel and color models:-

Pixels are the smallest individual element of an image. So, each pixel represents a part of the original image. It means, higher the pixel number- higher or much accurate representations of the actual picture. In colored images, they have three main color components(RGB- Red, Green, Blue), with pixel values of 0-255 for each pixel. So a pixel of (255, 255, 255) will represent a white and (0,0,0) means black. As the maximum number an 8-bit binary number can represent 255, is the maximum number we can go. As the base of binary-number is 2, we can convert the binary number into decimal very easily. Let, our binary number is 01010101, then its equivalent decimal number(base 10) will be: 26 +24 + 22 + 20 = 64 + 16 + 4 + 1 = 85



Least Significant Bit Steganography:-

Least Significant Bit (LSB) is a technique in which the last bit of each pixel is modified and replaced with the secret message's data bit. The idea behind LSB embedding is that if we change the last one or two bit value of a pixel, there won%u2019t be much visible change in the color. For example, 0 is black. Changing the value to 1 won make much of a difference since it is still black, just a lighter shade.

From the above image it is clear that, if we change MSB it will have a larger impact on the final value but if we change the LSB the impact on the final value is minimal, thus we use least significant bit steganography

How LSB technique works?

Each pixel contains three values which are Red, Green, Blue, these values range from 0 to 255, in other words, they are 8-bit values. Let's take an example of how this technique works, suppose you want to hide the message "hi" into a 4x4 image which has the following pixel values: [(225, 12, 99), (155, 2, 50), (99, 51, 15), (15, 55, 22),(155, 61, 87), (63, 30, 17), (1, 55, 19), (99, 81, 66),(219, 77, 91), (69, 39, 50), (18, 200, 33), (25, 54, 190)]

Using the ASCII Table, we can convert the secret message into decimal values and then into binary: 0110100 0110101.Now, we iterate over the pixel values one by one, after converting them to binary, we replace each least significant bit with that message bits sequentially (e.g 225 is 11100001, we replace the last bit, the bit in the right (1) with the first data bit (0) and so on).This will only modify the pixel values by +1 or -1 which is not noticeable at all. The resulting pixel values after performing LSBS are as shown below: [(224, 13, 99),(154, 3, 50),(98, 50, 15),(15, 54, 23),(154, 61, 87),(63, 30, 17),(1, 55, 19),(99, 81, 66),(219, 77, 91),(69, 39, 50),(18, 200, 33),(25, 54, 190)]

Caesar Cipher in Cryptography:-

The Caesar Cipher technique is one of the earliest and simplest method of encryption technique. It's simply a type of substitution cipher, i.e., each letter of a given text is replaced by a letter some fixed number of positions down the alphabet. For example with a shift of 1, A would be replaced by B, B would become C, and so on. The method is apparently named after Julius Caesar, who apparently used it to communicate with his officials.

Thus to cipher a given text we need an integer value, known as shift which indicates the number of position each letter of the text has been moved down.

The encryption can be represented using modular arithmetic by first transforming the letters into numbers, according to the scheme, A = 0, B = 1,…, Z = 25. Encryption of a letter by a shift n can be described mathematically as:-

(Encryption Phase with shift n)

$$\mathbf{E_n(x) = (x + n) \bmod 26}.$$

(Decryption Phase with shift n)

$$\mathbf{Dn(x) = (xi - n) \bmod 26}$$

Algorithm for Caesar Cipher:

Input:

1. Text.
2. An Integer between 0-25 denoting the required shift.

Procedure:

1. Traverse the given text one character at a time .
2. For each character, transform the given character as per the rule, depending on whether we're encrypting or decrypting the text.
3. Return the new string generated.

Ex;-

```
Text: ATTACKATONCE
Shift: 4
Cipher: EXXEGOEXSRGI
```

Rail Fence Cipher in Cryptography:-

In the rail fence cipher, the plaintext is written downwards diagonally on successive "rails" of an imaginary fence, then moving up when the bottom rail is reached, down again when the top rail is reached, and so on until the whole plaintext is written out. The ciphertext is then read off in rows.

Encryption

In a transposition cipher, the order of the alphabets is re-arranged to obtain the cipher-text.

1. In the rail fence cipher, the plain-text is written downwards and diagonally on successive rails of an imaginary fence.
2. When we reach the bottom rail, we traverse upwards moving diagonally, after reaching the top rail, the direction is changed again. Thus the alphabets of the message are written in a zig-zag manner.
3. After each alphabet has been written, the individual rows are combined to obtain the cipher-text.

For example:-

```
Input : "attack at once"  Key = 2
```

```
a t c a o c
```

```
t a k t n e
```

```
Output: atcaoctaktne
```

Decryption

As we've seen earlier, the number of columns in rail fence cipher remains equal to the length of plain-text message. And the key corresponds to the number of rails.

1. Hence, rail matrix can be constructed accordingly. Once we've got the matrix we can figure-out the spots where texts should be placed (using the same way of moving diagonally up and down alternatively).
2. Then, we fill the cipher-text row wise. After filling it, we traverse the matrix in zig-zag manner to obtain the original text

```
For ex
```

For example:-

```
Input : " atcaoctaktne "  Key = 2
a t c a o c
t a k t n e
Output : attack at once
```

The_secret_v2

Based on the techniques and concepts discussed earlier, we have built a The_secret_v2 program, which allows users to hide and encrypt their confidential data, which is represented by text or images, and in this section we will explain how the programs work and their logic.

The_secret_V2 program performs 6 main functions

1. **Encrypting a specific text using two algorithms, Caesar cipher and rail Fence cipher**
2. **hide image in image**
3. **hide text in image**
4. **Extract image from image**
5. **Extract text from image**
6. **Decrypt text using two algorithms, Caesar cipher and rail Fence cipher.**

The main idea is to encrypt and decrypt a text using rail fence and Caesar cipher

```
Proof of concept

Example using fence and Caesar cipher

First, we have the text that we want to encrypt it

THE PLAINT TEXT : ModyMorkosKamil

fence cipher process: split the text in 2 rows

M d m r o k m l
o y o k s a i x
```

THE CIPHER IS: MdMroKmloyoksaix

then encrypt CIPHER by using Caesar cipher -> K=2

our guild: abcdefghijklmnopqrstuvwxyz

THE FINAL CIPHER IS : OfOtqMonqaqmuckz

Secondly, we have the cipher that we want to decrypt it using caser cipher

THE CIPHER TEXT : OfOtqMonqaqmuckz

our guild: abcdefghijklmnopqrstuvwxyz

THE PLAINT TEXT : MdMroKmloyoksaix

fence chipher process: divid the cipher text on two rows in 2 rows

M d M r o K m l
o y o k s a i x

then Take a character from the first line once and from the second line once

THE FINAL PLAINT TEXT: ModyMorkosKamilx

caser cipher In python code

```python
def KeyUpperCase(char,K):
    return chr((ord(char)-65 +K)% 26 + 65)
def KeyLowerCase(char,K):
    return chr((ord(char)-97 +K)% 26 + 97)
def keyNumber(char,K):
    return chr((ord(char)-48 +K)% 10 + 48)

# Keys for decrypt
def DKeyUpperCase(char,K):
    return chr((ord(char)-65 -K)% 26 + 65)
def DKeyLowerCase(char,K):
    return chr((ord(char)-97 -K)% 26 + 97)
def DkeyNumber(char,K):
    return chr((ord(char)-48 -K)% 10 + 48)

#caser cipher

def encrypt_CC(data,K):
    s=''
    for i in data:
        if ord(i) >= ord('A') and ord(i) <= ord('Z'):
            s += KeyUpperCase(i,K)
        elif ord(i) >= ord('a') and ord(i) <= ord('z'):
            s += KeyLowerCase(i,K)
        elif ord(i) >= ord('0') and ord(i) <= ord('9'):
            s+= keyNumber(i,K)
        else:
            s+=i
    return s
def decrypt_CC(data,K):

    s=''
    for i in data:
        if ord(i) >= ord('A') and ord(i) <= ord('Z'):
            s += DKeyUpperCase(i,K)
        elif ord(i) >= ord('a') and ord(i) <= ord('z'):
            s += DKeyLowerCase(i,K)
        elif ord(i) >= ord('0') and ord(i) <= ord('9'):
            s+= DkeyNumber(i,K)
        else:
            s+=i
    return s
```

Rail fence in python code

```
def encode_RFC(x):
    if len(x) %2 != 0:
        x+='x'

    C = len(x)
    p1 = ''
    p2 = ''
    i=0
    while i< C-1:
        p1+=x[i]
        p2+=x[i+1]
        i+=2
    c=p1+p2
    return c
```

```
def decode_RFC(c):
    D = len(c)
    INC=int(0.5*D)
    plain=''
    p1=''
    p2=''
    for i in range(INC):
        p1+=c[i]
        p1+=c[int(i+INC)]
    plain=p1
    return plain
```

Caser and rail fence together

```
x='ModyMorkosKamil'
print(f'''
\t\t\t\t\t Proof of concept

\t\t\t\t Example using fence and ceasar chipher

First, we have the text that we want to encrypt it

THE PLAINT TEXT : {x}

fence chipher process: split the text in 2 rows

M d m r o k m l
o y o k s a i x

THE CIPHER IS: {encode_RFC(x)}

then encrypt CIPHER by using ceaser chipher -> K=2

our guild: abcdefghijklmnopqrstuvwxyz

THE FINAL CIPHER IS : {encrypt_CC(encode_RFC(x),2)}

Secondly, we have the cipher that we want to decrypt it using caser cipher

THE CIPHER TEXT : {CI}

our guild: abcdefghijklmnopqrstuvwxyz

THE PLAINT TEXT : {decrypt_CC(CI,2)}

fence chipher process: divid the cipher text on two rows in 2 rows

M d M r o K m l
o y o k s a i x

then Take a character from the first line once and from the second line once

THE FINAL PLAINT TEXT: {decode_RFC(decrypt_CC(CI,2))}
```

The main idea to hide text in image

Each keyboard input has a value in an ASCII table that can be converted to b inary. From here, since image pixels consist of decimal numbers and can be converted to binary, and from here we can hash the input like a specific lette r and save its value in the first two bits in the pixel RGB. This is a way to hi de text in An image and when extracting, we pull the bits again and collect it in an organized way and convert it to ASCII again

Thy python code ASCII to binarry

```
def to_bin(data):
    """Convert `data` to binary format as string"""
    if isinstance(data, str):
        return ''.join([ format(ord(i), "08b") for i in data ])
    elif isinstance(data, bytes) or isinstance(data, np.ndarray):
        return [ format(i, "08b") for i in data ]
    elif isinstance(data, int) or isinstance(data, np.uint8):
        return format(data, "08b")
    else:
        raise TypeError("Type not supported.")
```

Hide binary into image pixels

```
]: def hideData(image, secret_massage):
    n_bytes = image.shape[0]*image.shape[1] *3 // 8
    print('Maximum bytes to ebcode:',n_bytes)

    if len(secret_massage) > n_bytes:
        raise ValueError("Massage bigger than image")

    secret_massage += "######"
    data_index = 0

    binary_secret_msg = to_bin(secret_massage)

    data_len = len(binary_secret_msg)


    for values in image:
        for pixiel in values:
            r,g,b = to_bin(pixiel)
            if data_index < data_len:
                pixiel[0] = int(r[:-1] + binary_secret_msg[data_index],2)
                data_index +=1
            if data_index < data_len:
                pixiel[1] = int(g[:-1] + binary_secret_msg[data_index],2)
                data_index +=1
            if data_index < data_len :
                pixiel[2] = int(b[:-1] + binary_secret_msg[data_index],2)
                data_index += 1
            if data_index >= data_len:
                break
    return image
```

Extract binary from image pixels

```
: def showData(image):
    binary_data = ""
    for values in image:
        for pixiel in values:
            r,g,b = to_bin(pixiel)
            binary_data += r[-1]
            binary_data += g[-1]
            binary_data += b[-1]

    all_bytes = [binary_data[i: i+8] for i in range(0,len(binary_data),8)]
    decoded_data = ""
    for byte in all_bytes:
        decoded_data += chr(int(byte ,2))
        if decoded_data[-5:]=="######":
            break
    return decoded_data[:-5]
```

The ASCII Character Set

ASCII stands for the "American Standard Code for Information Interchange". It was designed in the early 60's, as a standard character set for computers and electronic devices .ASCII is a 7-bit character set containing 128 characters. It contains the numbers from 0-9, the upper and lower case English letters from A to Z, and some special characters. The character sets used in modern computers, in HTML, and on the Internet, are all based on ASCII. The following tables list the 128 ASCII characters and their equivalent number.

The main idea to hide image in image

n order to hide a picture within a picture, the secret picture must be less than or equal to the size of the main picture. The method of hiding an image within an image depends on taking the last two or three bits of the secret image's pixels using the MSB method, and saving them in the base image using the LSB method.The image will appear in color slightly different from the original image

What the below functions do are self explanatory from their names.
They use python's bit operations to return a required value.
We will use these functions in later part of the script.
If you don't understand their use now don't worry we will see it below.

```
def remove_n_least_significant_bits(value, n):
    value = value >> n
    return value << n

def get_n_least_significant_bits(value, n):
    value = value << MAX_BIT_VALUE - n
    value = value % MAX_COLOR_VALUE
    return value >> MAX_BIT_VALUE - n

def get_n_most_significant_bits(value, n):
    return value >> MAX_BIT_VALUE - n

def shit_n_bits_to_8(value, n):
    return value << MAX_BIT_VALUE - n
```

Now we will write the encode function that will do the encoding.
Here we are assuming that the images are of the same size.
I have written required comments to explain the below code.

```
def encode(image_to_hide, image_to_hide_in, n_bits):

    width, height = image_to_hide_in.size

    #.load() returns the "pixel_access" object that has the data(matrix) of the pixels.
    hide_image = image_to_hide.load()
    hide_in_image = image_to_hide_in.load()

    #this will store the values of each individual pixel as a matrix.
    data = []

    #looping the hide_image object.

    for y in range(height):
        for x in range(width):

            # (107, 3, 10)
            # most sig bits
            #print(hide_image[x,y]) #you can uncomment this to see the pixel values in r,g,b form.
            try:
                #the value of n can be 1 or 2 and you won't see much difference in the encoded image.
                #gets n most significant bits of r,g,b values of image to hide.
                r_hide, g_hide, b_hide= hide_image[x,y]
                r_hide = get_n_most_significant_bits(r_hide, n_bits)
                g_hide = get_n_most_significant_bits(g_hide, n_bits)
                b_hide = get_n_most_significant_bits(b_hide, n_bits)

                # remove least n significant bits of image to hide in so we can store
                # the n most significant bits in that place.

                r_hide_in, g_hide_in, b_hide_in= hide_in_image[x,y]
                r_hide_in = remove_n_least_significant_bits(r_hide_in, n_bits)
                g_hide_in = remove_n_least_significant_bits(g_hide_in, n_bits)
                b_hide_in = remove_n_least_significant_bits(b_hide_in, n_bits)
```

```
            #incase of exception it will show the reason.
            except Exception as e:
                print(e)

    #return an Image object from the above data.
    return make_image(data, image_to_hide.size)


#takes image to decode and n_bits as parameters.
```

Now we will write decode function and run it.

comment the part that you wrote for the running the encode function.

The decode function should look something like this.

```
def decode(image_to_decode, n_bits):
    width, height = image_to_decode.size
    encoded_image = image_to_decode.load()

    #matrix that will store the extracted pixel values from the encoded Image.
    data = []

    #looping through the encoded Image.
    for y in range(height):
        for x in range(width):

            #gets rgb values of encoded image.
            r_encoded, g_encoded, b_encoded = encoded_image[x,y]

            #get n least significant bits for each r,g,b value of the encoded image
            r_encoded = get_n_least_significant_bits(r_encoded, n_bits)
            g_encoded = get_n_least_significant_bits(g_encoded, n_bits)
            b_encoded = get_n_least_significant_bits(b_encoded, n_bits)

            #shifts the n bits to right so that they occupy a total of 8 bit spaces.
            #like if there 10 are the bits then shifting them would look like 10000000
            #this would ofcourse be converted to an int as per python's bit operations.

            r_encoded = shit_n_bits_to_8(r_encoded, n_bits)
            g_encoded = shit_n_bits_to_8(g_encoded, n_bits)
            b_encoded = shit_n_bits_to_8(b_encoded, n_bits)

            data.append((r_encoded, g_encoded, b_encoded))

    return make_image(data, image_to_decode.size)
```

The main:-

```
def THE_secret_V2():
    n_bits = 2
    print('''        \t   \t      THE_secret_V2 pro
    This program helps you to encrypt and hide your sensitive data in a image
    and also to decrypt and extract your  data form the image
    the secrit data divided into secret image and secret data in one image''')
    print("\n")
    cmd = int(input("THE_secret_V2 pro \n 1.Encode the data \n 2.Decode the data \n Your input is: "))
    if cmd==1:
        print('''\nthe Encode data work flow is
        1-choose your main image to hide image and test data
        2-choose your secret image that you want to hide
        3-choose your secret text that you want to hide (encrypted by two algorisms)
        4-save your final image that has image and test\n''')
        The_main_image_path =input("Enter The main image name (with (jpg or png) extension)\n")
        The_secret_image_path = input("Enter the secre image name (with jpg extension)\n")
        The_main_image = Image.open(The_main_image_path)
        The_secret_image = Image.open(The_secret_image_path)
        The_secret_image=The_secret_image.resize(The_main_image.size)
        encoded_image_path = input("Enter the encoded image name to save (with png extension)\n")
        encode(The_secret_image, The_main_image, n_bits).save(encoded_image_path)
        encode_text(encoded_image_path)


    elif cmd==2:
        print('''the Decode data work flow is
        1-choose your main image that has secret image/text
        2-extract the secret text from image (decrypted by two algorisms)
        3-extract the secret image from image ''')
        encoded_image_path = input("Enter the encoded image name to decoded ")
        decoded_image_path = input("Enter the decoded image name to save ")
        D = decode_text(encoded_image_path)
        K = int(input('enter the right Key to decrypt '))
        C=decrypt_CC(D,K)
        P=decode_RFC(C)
        print("the massage depend on yours K\n",P)

        image_to_decode = Image.open(encoded_image_path)
        decode(image_to_decode, n_bits).save(decoded_image_path)
        Image.open(decoded_image_path)
    else :
        print(Exception("Enter correct input"))
```

The output for encoding:-

```
In [59]: THE_secret_V2()
                                        THE_secret_V2 pro
            This program helps you to encrypt and hide your sensitive data in a image
            and also to decrypt and extract your  data form the image
            the secrit data divided into secret image and secret data in one image


          THE_secret_V2 pro
           1.Encode the data
           2.Decode the data
           Your input is: 1

          the Encode data work flow is
                  1-choose your main image to hide image and test data
                  2-choose your secret image that you want to hide
                  3-choose your secret text that you want to hide (encrypted by two algorisms)
                  4-save your final image that has image and test

          Enter The main image name (with (jpg or png) extension)
          account.png
          Enter the secre image name (with jpg extension)
          mody.jpg
          Enter the encoded image name to save (with png extension)
          enc.png
          the shape of image is  (1000, 1000, 3)
          Enter data to hide ModyMorkosKamil
          Enter the key for caser cipher5
          your massage encrypted as :
           RiRwtPrqtdtpxfnc
          save image as (png ):
          enc.png
          Maximum bytes to ebcode: 375000
```

The output for decoding:-

```
                              THE_secret_V2 pro
    This program helps you to encrypt and hide your sensitive data in a image
    and also to decrypt and extract your  data form the image
    the secrit data divided into secret image and secret data in one image


THE_secret_V2 pro
 1.Encode the data
 2.Decode the data
 Your input is: 2
the Decode data work flow is
        1-choose your main image that has secret image/text
        2-extract the secret text from image (decrypted by two algorisms)
        3-extract the secret image from image
Enter the encoded image name to decoded enc.png
Enter the decoded image name to save dec.png

 Decoded massage is :
 RiRwtPrqtdtpxfnc
enter the right Key to decrypt 5
the massage depend on yours K
 ModyMorkosKamilx
```

The main image:-

The original image:-

The image after extracting:-