# Abstract

This Python code utilizes the Tkinter library to create a user-friendly graphical user interface (GUI) application. The purpose of this application is to encrypt and decrypt personal information in images. Users are able to input their name, Turkish ID number (TC), and upload a grayscale image for encryption. The encryption process involves a basic algorithm that utilizes XOR operations with a randomly generated password. The encrypted data is then embedded in the image by manipulating the pixel values.

To convert the image into a format suitable for encryption, the script employs the Pillow (PIL) library. This library allows the image to be converted into a NumPy array, where specific pixel values represent binary information. The application also supports image decryption, where the encrypted data is extracted from the image and the personal information is displayed in its decrypted form.

The encryption algorithm involves transforming characters into binary and encoding them as specific pixel values in the image. The decryption process reverses this, extracting the binary data and deciphering it using the same password.

Additionally, the code includes a function to generate a random password, as well as auxiliary functions to convert binary data to ASCII and extract binary information from specific pixel values in the image.

In summary, this project provides a user-friendly interface for encrypting and decrypting personal information in images. It offers a straightforward application of basic encryption techniques.

# Introduction

This project is centered around the implementation of a data transmission system that utilizes images as the medium and incorporates encryption and decryption processes. The primary objective of this system is to securely store and retrieve personal information, such as names, surnames, and identification numbers.

To achieve this, our project makes use of the Python programming language and the Tkinter library to create a user-friendly graphical interface. The encryption and decryption functionalities are implemented using the Python Imaging Library (PIL), which allows for the manipulation of image data. During the encryption process, sensitive information is concealed within the pixels of an image. Conversely, the decryption process extracts and reconstructs the original data.

The user interface (UI) of our system provides a simple and intuitive experience for users. They are able to input their personal information, including their name, surname, and identification number, for encryption. Additionally, the system allows users to upload an image that will be used in the encryption process.

The encryption process begins by generating a random password. This password is then used to encrypt the provided personal information using the XOR technique. The encrypted data is then embedded into the pixel values of the image, ensuring a secure transmission.

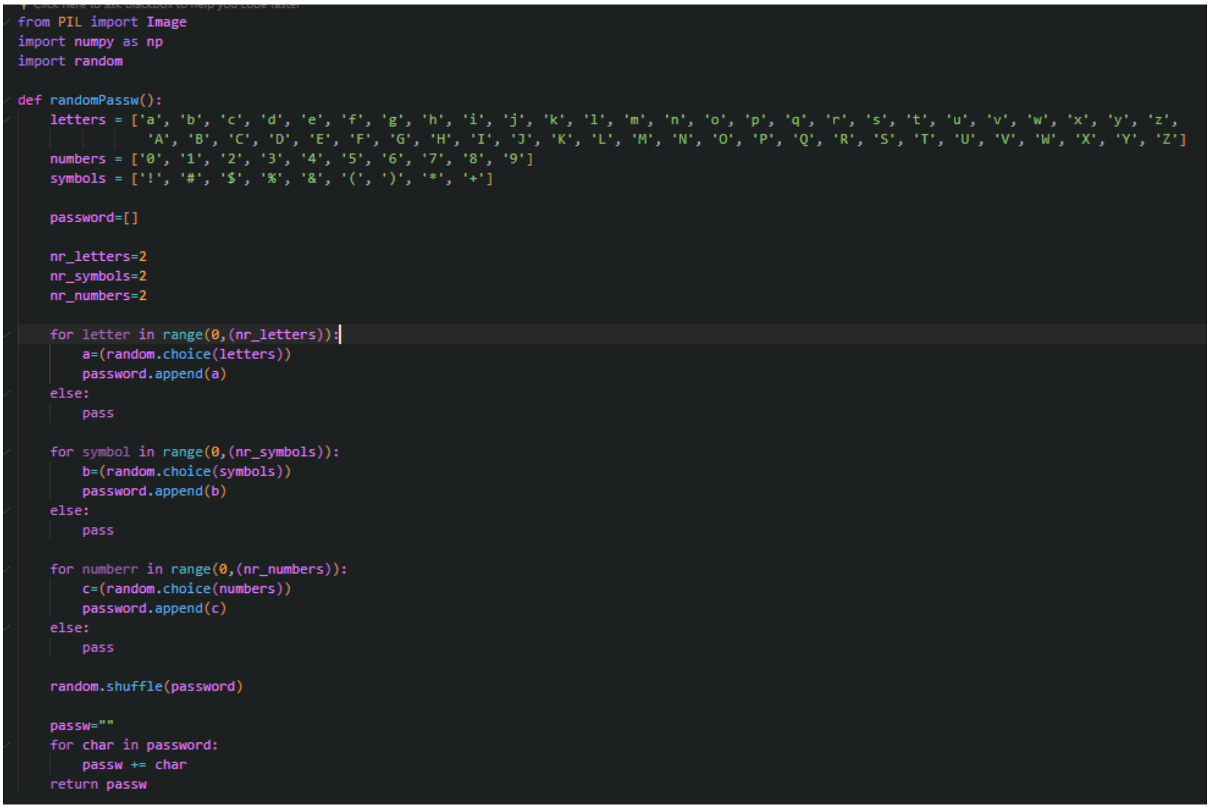
On the other hand, the decryption process enables the extraction of the encrypted data from the image using the generated password. This process ensures the recovery of the original information without compromising its security.

To enhance the overall security of the system, a secure password generation algorithm has been implemented. This algorithm creates a complex password by combining uppercase and lowercase letters, numbers, and symbols. By utilizing this algorithm, the system ensures that the generated passwords are difficult to guess or crack, further safeguarding the security of the transmitted data.

# Implementation

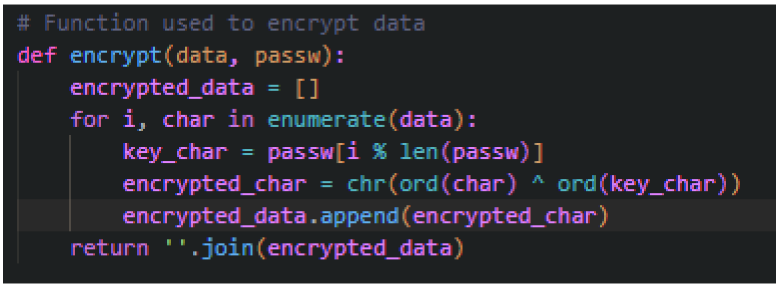
## *encryption.py*

### randomPassw() function;



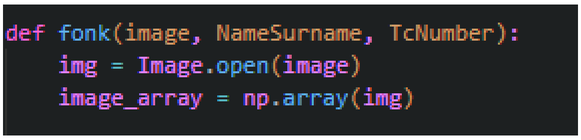
```
Click here to ask Blackbox to help you code faster
from PIL import Image
import numpy as np
import random

def randomPassw():
    letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z',
                'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']
    numbers = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
    symbols = ['!', '#', '$', '%', '&', '(', ')', '*', '+']

    password=[]

    nr_letters=2
    nr_symbols=2
    nr_numbers=2

    for letter in range(0,(nr_letters)):
        a=(random.choice(letters))
        password.append(a)
    else:
        pass

    for symbol in range(0,(nr_symbols)):
        b=(random.choice(symbols))
        password.append(b)
    else:
        pass

    for numberr in range(0,(nr_numbers)):
        c=(random.choice(numbers))
        password.append(c)
    else:
        pass

    random.shuffle(password)

    passw=""
    for char in password:
        passw += char
    return passw
```

### Encrypt() function;



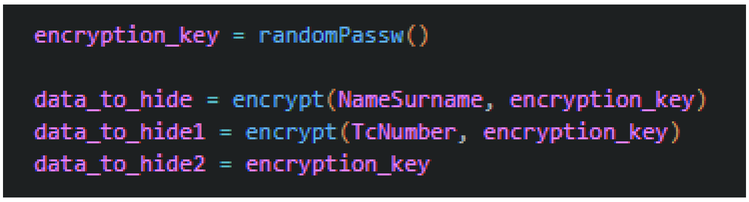
```
# Function used to encrypt data
def encrypt(data, passw):
    encrypted_data = []
    for i, char in enumerate(data):
        key_char = passw[i % len(passw)]
        encrypted_char = chr(ord(char) ^ ord(key_char))
        encrypted_data.append(encrypted_char)
    return ''.join(encrypted_data)
```

1) First we open image and with numpy library we make image pixel.



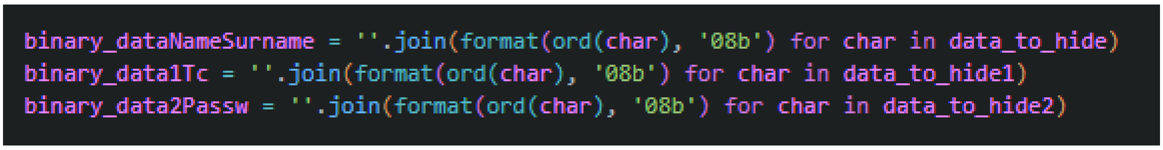
```
def fonk(image, NameSurname, TcNumber):
    img = Image.open(image)
    image_array = np.array(img)
```

2) Get special encryption key from randomPassw() for his image and encrypt the user information with this key.



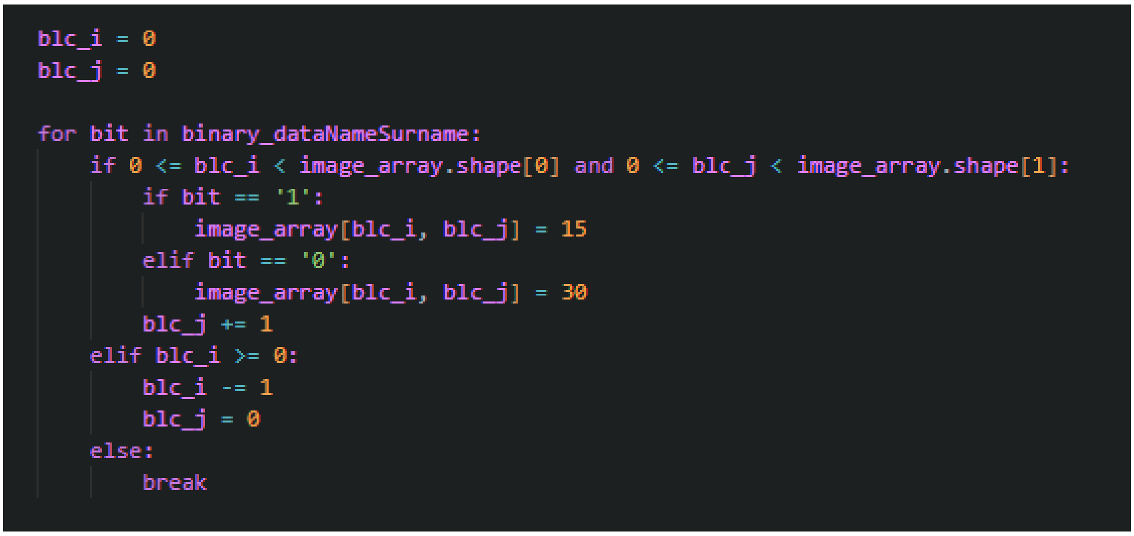
```
encryption_key = randomPassw()

data_to_hide = encrypt(NameSurname, encryption_key)
data_to_hide1 = encrypt(TcNumber, encryption_key)
data_to_hide2 = encryption_key
```

3) Transform this information to binary form.



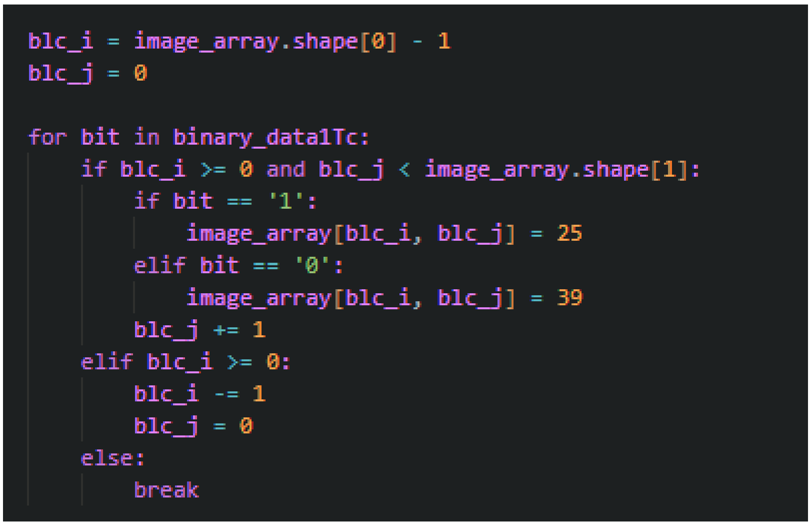
```
binary_dataNameSurname = ''.join(format(ord(char), '08b') for char in data_to_hide)
binary_data1Tc = ''.join(format(ord(char), '08b') for char in data_to_hide1)
binary_data2Passw = ''.join(format(ord(char), '08b') for char in data_to_hide2)
```

4) Embed name and surname into image, corner top left.



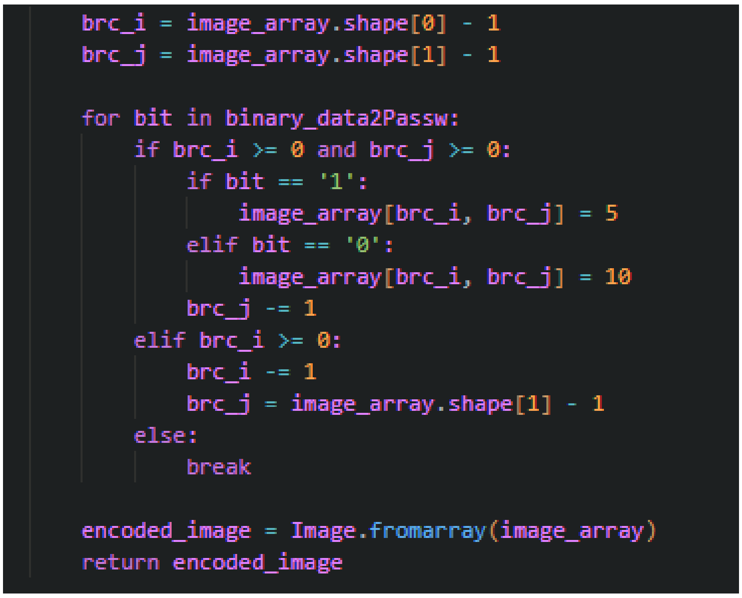
```
blc_i = 0
blc_j = 0

for bit in binary_dataNameSurname:
    if 0 <= blc_i < image_array.shape[0] and 0 <= blc_j < image_array.shape[1]:
        if bit == '1':
            image_array[blc_i, blc_j] = 15
        elif bit == '0':
            image_array[blc_i, blc_j] = 30
        blc_j += 1
    elif blc_i >= 0:
        blc_i -= 1
        blc_j = 0
    else:
        break
```

5) Embed TC number into image, left bottom.



```
blc_i = image_array.shape[0] - 1
blc_j = 0

for bit in binary_data1Tc:
    if blc_i >= 0 and blc_j < image_array.shape[1]:
        if bit == '1':
            image_array[blc_i, blc_j] = 25
        elif bit == '0':
            image_array[blc_i, blc_j] = 39
        blc_j += 1
    elif blc_i >= 0:
        blc_i -= 1
        blc_j = 0
    else:
        break
```

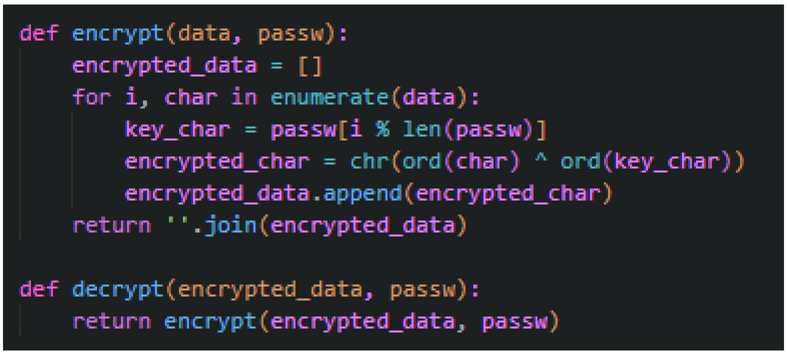
6) Embed password into img right bottom.



```
    brc_i = image_array.shape[0] - 1
    brc_j = image_array.shape[1] - 1

    for bit in binary_data2Passw:
        if brc_i >= 0 and brc_j >= 0:
            if bit == '1':
                image_array[brc_i, brc_j] = 5
            elif bit == '0':
                image_array[brc_i, brc_j] = 10
            brc_j -= 1
        elif brc_i >= 0:
            brc_i -= 1
            brc_j = image_array.shape[1] - 1
        else:
            break

    encoded_image = Image.fromarray(image_array)
    return encoded_image
```

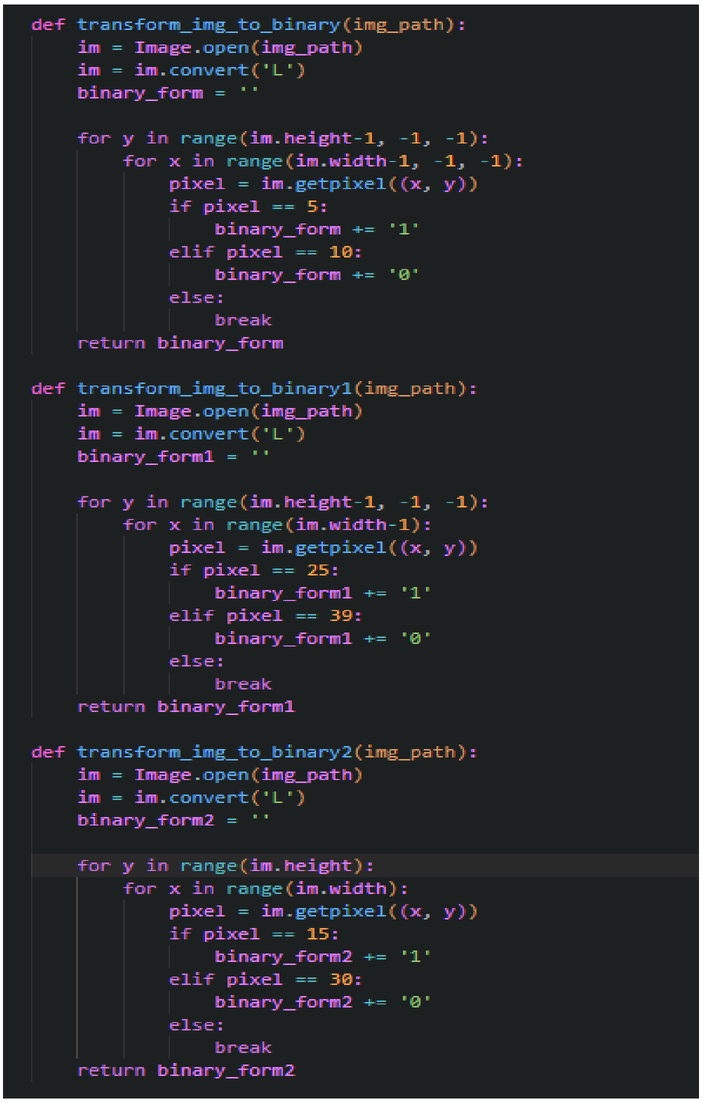
# ***decryption.py***

1) Encryption and decryption functions.



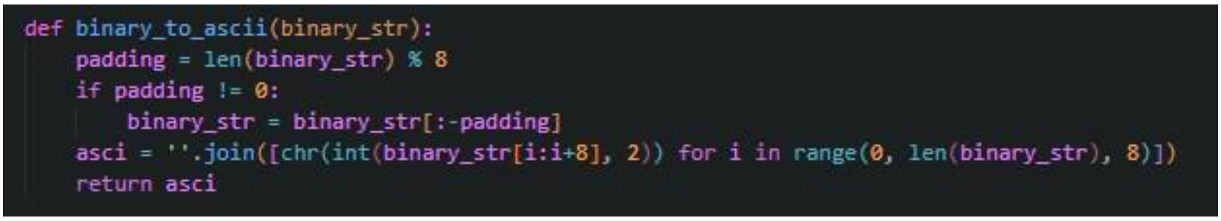
```python
def encrypt(data, passw):
    encrypted_data = []
    for i, char in enumerate(data):
        key_char = passw[i % len(passw)]
        encrypted_char = chr(ord(char) ^ ord(key_char))
        encrypted_data.append(encrypted_char)
    return ''.join(encrypted_data)

def decrypt(encrypted_data, passw):
    return encrypt(encrypted_data, passw)
```

2) Uproot information from image password, nameSurname and TC number.



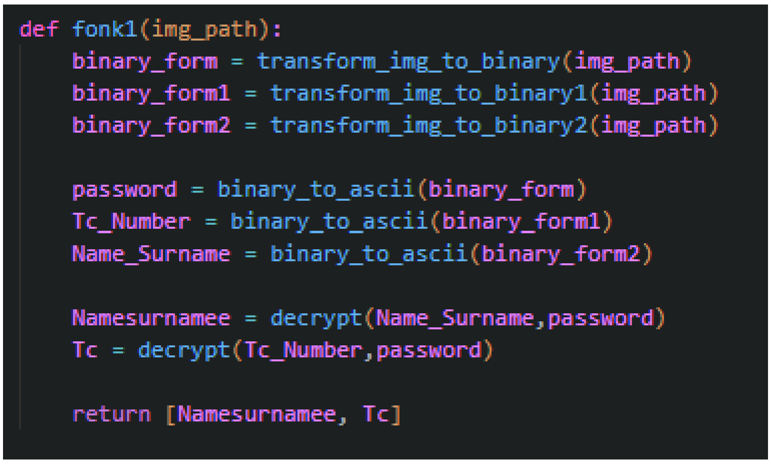
```python
def transform_img_to_binary(img_path):
    im = Image.open(img_path)
    im = im.convert('L')
    binary_form = ''

    for y in range(im.height-1, -1, -1):
        for x in range(im.width-1, -1, -1):
            pixel = im.getpixel((x, y))
            if pixel == 5:
                binary_form += '1'
            elif pixel == 10:
                binary_form += '0'
            else:
                break
    return binary_form

def transform_img_to_binary1(img_path):
    im = Image.open(img_path)
    im = im.convert('L')
    binary_form1 = ''

    for y in range(im.height-1, -1, -1):
        for x in range(im.width-1):
            pixel = im.getpixel((x, y))
            if pixel == 25:
                binary_form1 += '1'
            elif pixel == 39:
                binary_form1 += '0'
            else:
                break
    return binary_form1

def transform_img_to_binary2(img_path):
    im = Image.open(img_path)
    im = im.convert('L')
    binary_form2 = ''

    for y in range(im.height):
        for x in range(im.width):
            pixel = im.getpixel((x, y))
            if pixel == 15:
                binary_form2 += '1'
            elif pixel == 30:
                binary_form2 += '0'
            else:
                break
    return binary_form2
```

3) Transform binary to ascii function.



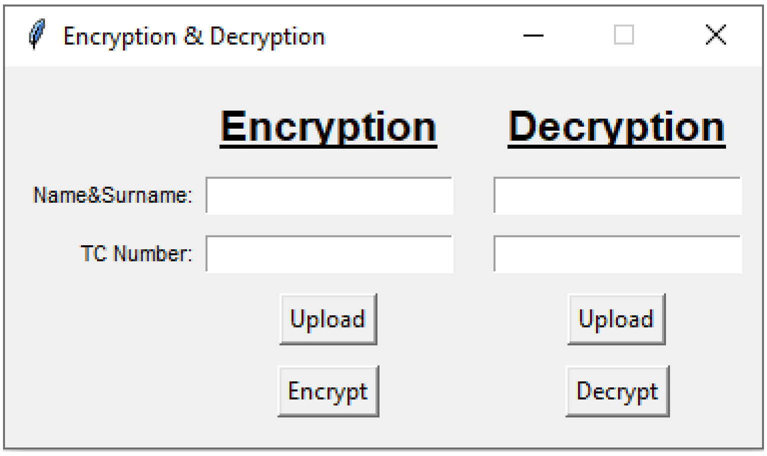
```
def binary_to_ascii(binary_str):
    padding = len(binary_str) % 8
    if padding != 0:
        binary_str = binary_str[:-padding]
    asci = ''.join([chr(int(binary_str[i:i+8], 2)) for i in range(0, len(binary_str), 8)])
    return asci
```

4) Our main function.



```
def fonk1(img_path):
    binary_form = transform_img_to_binary(img_path)
    binary_form1 = transform_img_to_binary1(img_path)
    binary_form2 = transform_img_to_binary2(img_path)

    password = binary_to_ascii(binary_form)
    Tc_Number = binary_to_ascii(binary_form1)
    Name_Surname = binary_to_ascii(binary_form2)

    Namesurnamee = decrypt(Name_Surname,password)
    Tc = decrypt(Tc_Number,password)

    return [Namesurnamee, Tc]
```

## ***User interface.py***



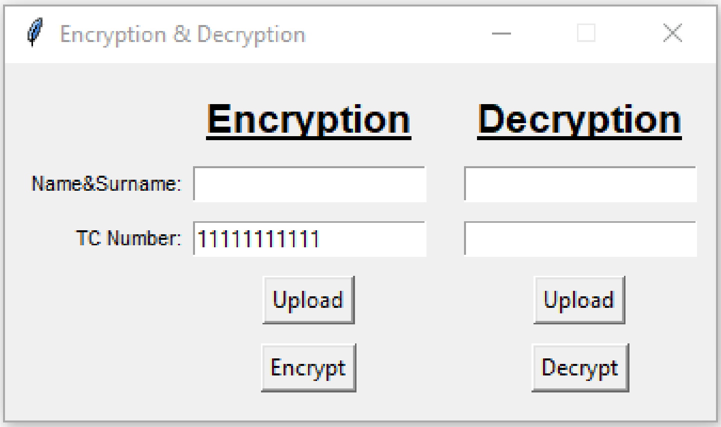
In Encryption part, user will fill Name&Surname and TC Number blanks and after that Upload an image which these information will be embed and press the Encrypt button. Later program will ask where to save this embedded image.

In Decryption part, user will upload embedded image from upload button and press Decrypt button. Atfer that the information that user embed will print out in the blanks.

# Character Limitations and Rules

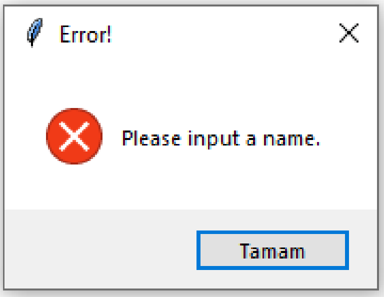
## Encryption Part

1. User has to input name and surname. Also Name Surname total limitation is 30 characters.



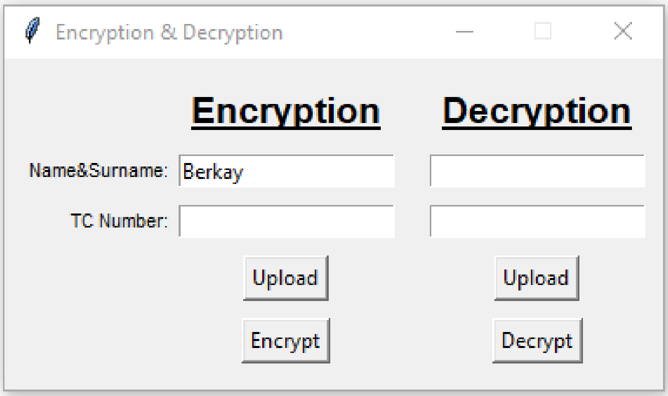
Name&Surname:

TC Number: 11111111111



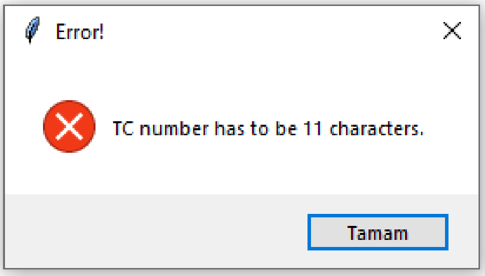
Please input a name.

2. User has to input TC Number exatcly 11 characters.

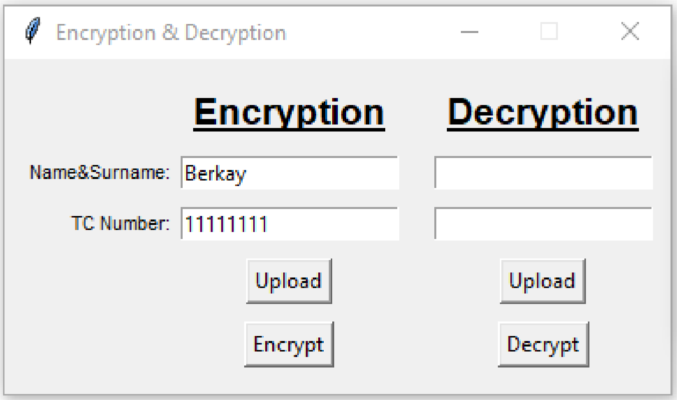


Name&Surname: Berkay

TC Number:

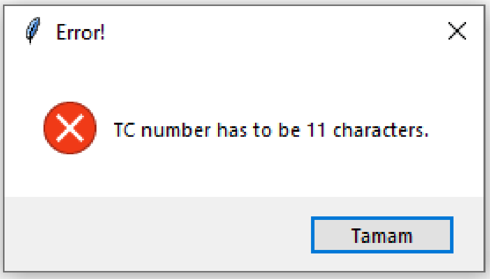


TC number has to be 11 characters.



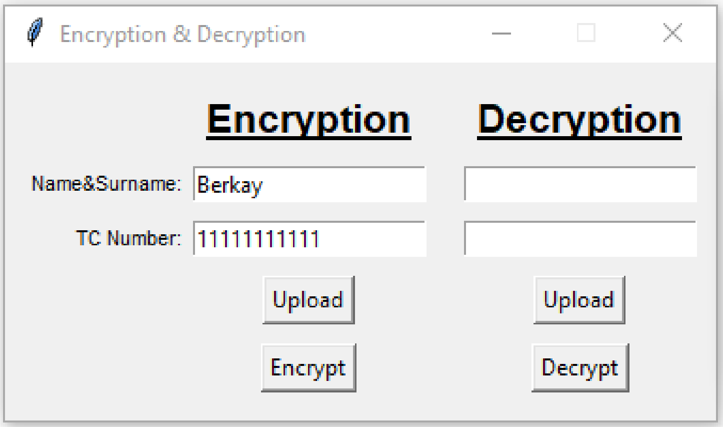
Name&Surname: Berkay

TC Number: 11111111



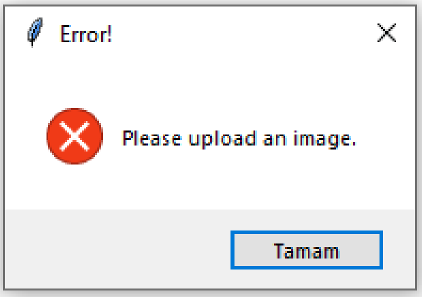
TC number has to be 11 characters.

3. User has to upload an image.



Name&Surname: Berkay

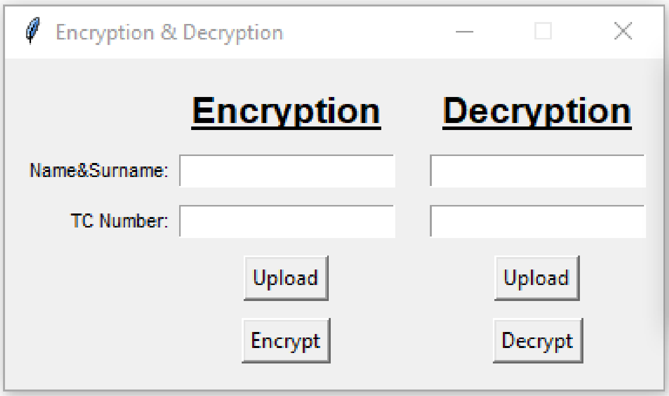
TC Number: 11111111111

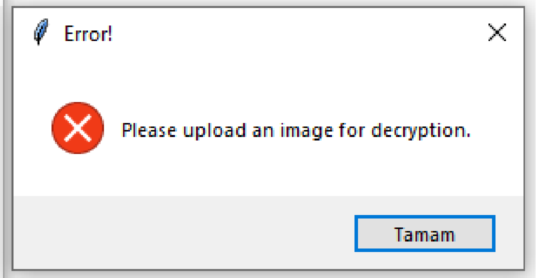


Please upload an image.

## Decryption Part

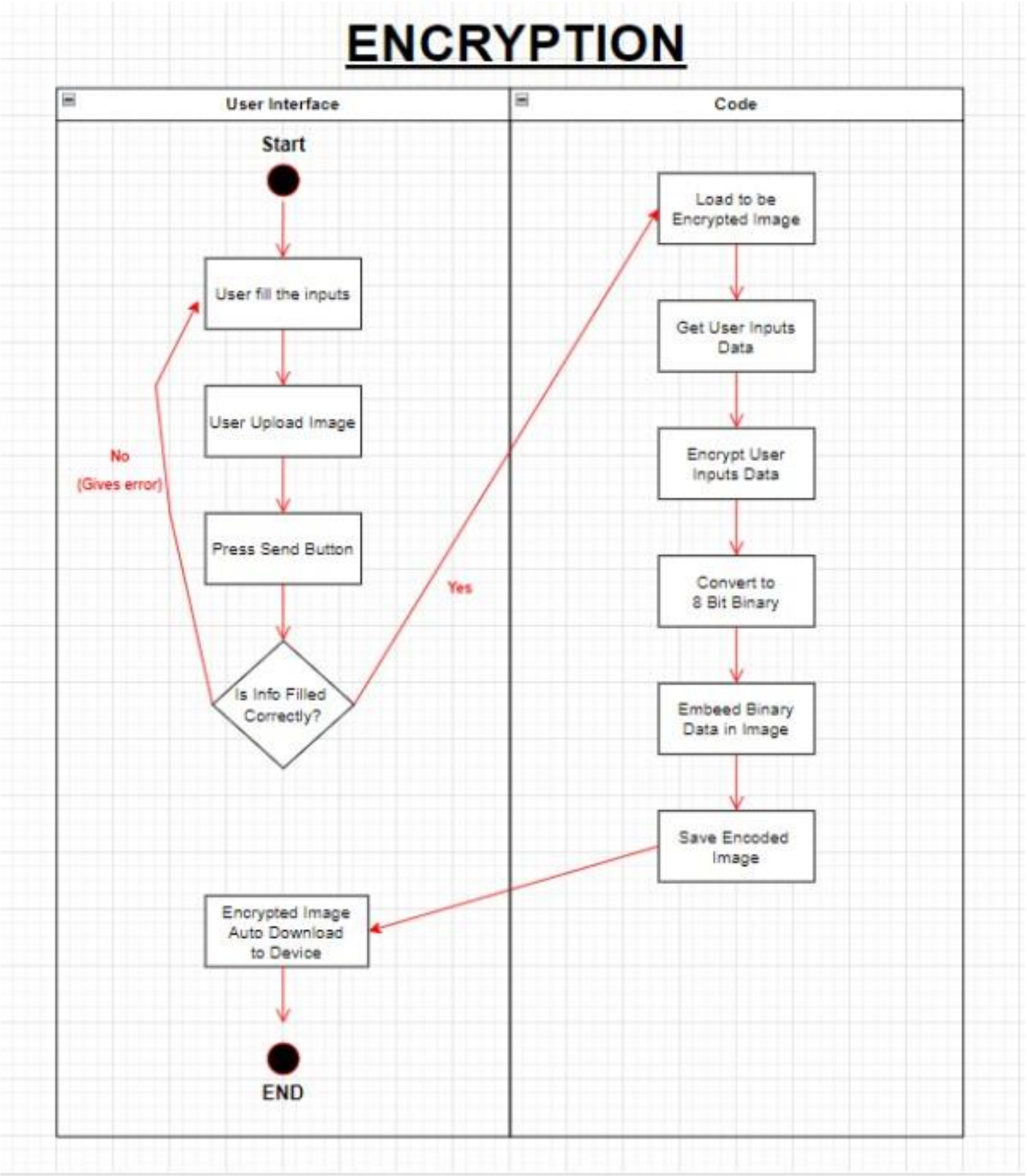
1. User has to upload encoded image.



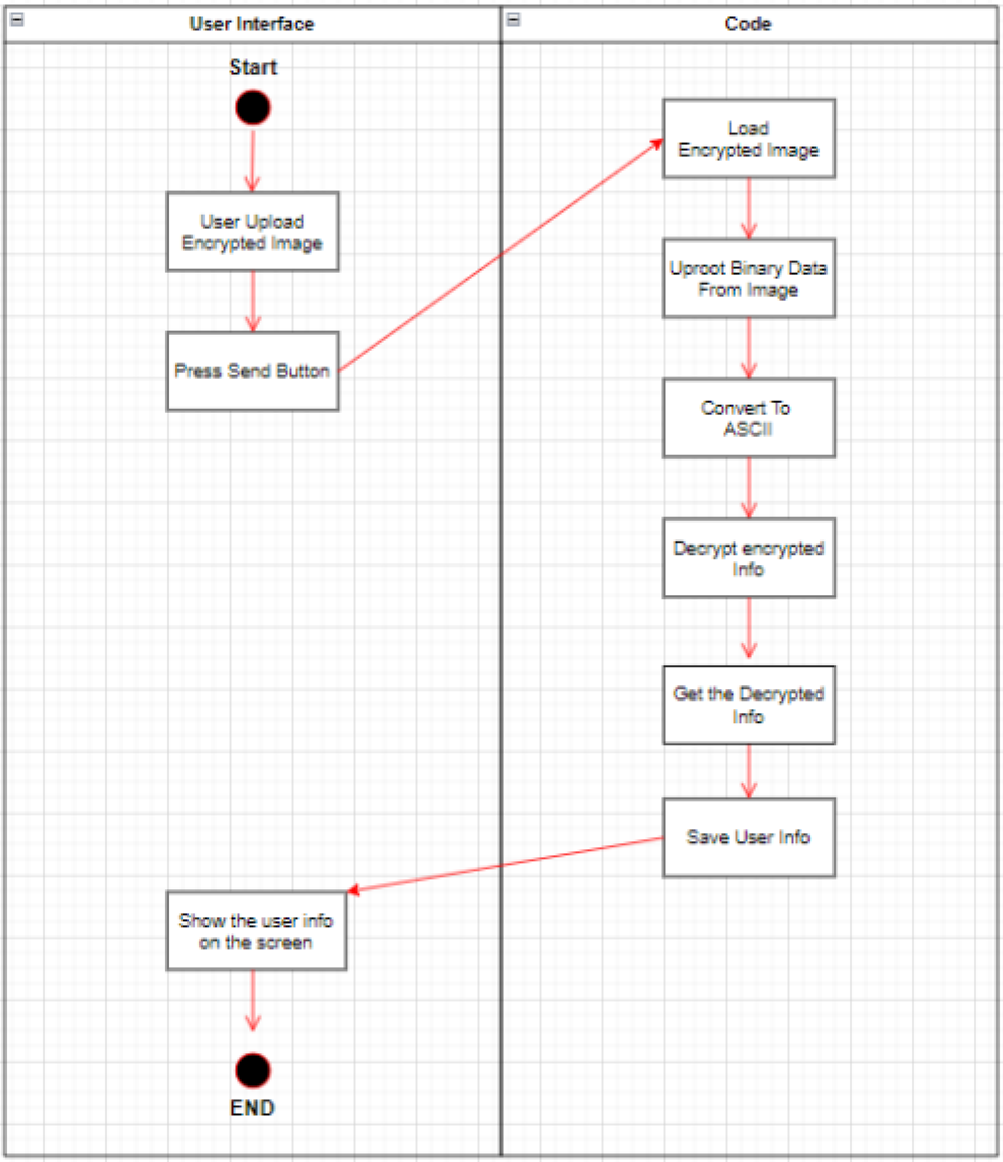


Please upload an image for decryption.

# Implementation with Diagrams



# DECRYPTION


User Interface
Code
Start
User Upload Encrypted Image
Press Send Button
Load Encrypted Image
Uproot Binary Data From Image
Convert To ASCII
Decrypt encrypted Info
Get the Decrypted Info
Save User Info
Show the user info on the screen
END

# Conclusion

To summarize, the successful implementation of an image-based data transmission system with encryption and decryption capabilities has resulted in an application that is both user-friendly and secure. This achievement was made possible through the integration of Tkinter for the graphical user interface, the Python Imaging Library (PIL) for image processing, and custom encryption algorithms to ensure the confidentiality of sensitive information.

The encryption process, which incorporates a random password generation algorithm, guarantees a high level of security during data transmission. By embedding encrypted data within image pixels, the system effectively establishes a robust and covert method of secure communication.

The decryption process efficiently extracts the encrypted data using the generated password, allowing for the retrieval of the original information without compromising security. The graphical user interface provides users with a simple and intuitive experience, enabling them to input their data, upload images, and execute encryption and decryption operations.

However, it is important to acknowledge certain limitations of the system. Currently, it only supports PNG images, and large data sizes may exceed the capacity of the chosen image. Additionally, while the implemented encryption technique offers a reasonable level of security, it may not be suitable for highly sensitive or classified information. In such cases, more advanced encryption methods would be necessary.

Looking ahead, there is potential for future improvements. These may include expanding support for different image formats and exploring advanced encryption techniques to address the identified limitations.

In conclusion, the image-based data transmission system exemplifies the successful integration of encryption principles, image processing techniques, and user interface design. It serves as a valuable tool for secure and private information exchange.

# References

1. Hazir, E. (2023, September 21). Python ile GUI Geliştirme | Örneklerle Tkinter - Enes HAZIR - Medium. Medium. https://eneshazr.medium.com/python-ile-gui-geli%C5%9Ftirme-%C3%B6rneklerle-tkinter-51ca1b82166b

2. Coding Ninjas. (n.d.). Coding Ninjas Studio. https://www.codingninjas.com/studio/library/how-do-you-get-the-logical-xor-of-two-variables-in-python

3. Kumar, A. (2022, July 25). What is XOR in Python? - Scaler Topics. Scaler Topics. https://www.scaler.com/topics/xor-in-python/

4. How to XOR two strings in Python. (n.d.). Stack Overflow. https://stackoverflow.com/questions/36242887/how-to-xor-two-strings-in-python
5. XOR RGB Image Decryption in Python. (n.d.). Stack Overflow.

   https://stackoverflow.com/questions/53754959/xor-rgb-image-decryption-in-python
6. NumPy: the absolute basics for beginners — NumPy v1.26 Manual. (n.d.). https://numpy.org/doc/stable/user/absolute_beginners.html

7. GeeksforGeeks. (2021, July 3). Python Pillow a fork of PIL. https://www.geeksforgeeks.org/python-pillow-a-fork-of-pil/

8. How do you get the logical xor of two variables in Python? (n.d.). Stack Overflow. https://stackoverflow.com/questions/432842/how-do-you-get-the-logical-xor-of-two-variables-in-python

9. GeeksforGeeks. (2023, July 31). Introduction to Tkinter. https://www.geeksforgeeks.org/introduction-to-tkinter/

10. Abba, I. V. (2023, May 8). xor.py – How the Python XOR Operator Works. freeCodeCamp.org. https://www.freecodecamp.org/news/xor-py-how-the-python-xor-operator-works/