Mini Project

“ENCRYPTION - DECRYPTION”

Submitted in partial fulfillment of the
requirements of the degree

BACHELOR OF ENGINEERING IN
COMPUTER ENGINEERING
By

Sahil A. Sawant B/17
Aditya P. Shinde B/25
Vinayak V. Utekar B/42
Group Number : **03**

Supervisor
Dr. Poulami Das
Department of Computer Engineering
K. C. College of Engineering and Management Studies and Research,
Thane- 400 603.
University of Mumbai

(AY 2020-21)

CERTIFICATE

This is to certify that the Mini Project Project entitled “**“ENCRYPTION - DECRYPTION”** ” is a bonafide work of Sahil A. Sawant B/17, Aditya P. Shinde B/25, Vinayak V. Utekar B/32 submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of “Bachelor of Engineering” in “Computer Engineering”.

(Dr. POULAMI DAS ROY)

Supervisor

(Prof. Mandar A. Ganjapurkar)
Head of Department

(Dr. Vilas Nitnaware)
Principal

Mini Project Approval

This Mini Project entitled "“**ENCRYPTION - DECRYPTION**” " by Sahil Sawant B/17, Aditya Shinde B/25, Vinayak Utekar B/32 is approved for the degree of Bachelor of Engineering in Computer Engineering.

Engineering Examiners

1..
(Internal Examiner Name & Sign)

2..
(External Examiner name & Sign)

Date : 30/04/2022

Place : Mumbai

Contents

ACKNOWLEDGEMENT

No project is ever complete without the guidance of those experts who have already traded their past before and hence became master of it and as a result, our mentor. So we would like to take this opportunity to take all those individuals who have helped us in visualizing this project.

The guidance of **Dr. POULAMI DAS ROY** played a great role in our research work. Her guidance helped us in finding relevant information about our topic. We are grateful to get an opportunity to present our work to everyone. We would like to express our gratitude to the '**K.C. College of Engineering and Management studies & Research**'' as well as our Head of Department, professor **"Mandar Ganjapurkar"** for promoting students to express their ideas and research. Our sincere vote of thanks goes to our college Principal, "**Dr. Vilas Nitnaware"** for believing in the work of their students and pushing our limits to do better in our field of study.

Abstract

Encoder-Decoder – Secure your Information by Encoding the messages. Encoding is the process that transforms the text or information to the unrecognizable form and decryption is the process to convert the encrypted message into its original form. The objective of this project is to encode and decode messages using Caesar Cipher and Steganography.

Message encoding is the process to first convert the original text to the random and meaningless text called ciphertext. This process is called encoding. Decoding is the process to convert that ciphertext to the original text. While in Steganography the existence of the text is hidden in normal unsecure image until it is decoded.

In this project, users have to enter the message and decide whether to encode or decode. To build this project we have used the basic concept of python, Tkinter and Stegano.

Tkinter : Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Steganography is the technique of hiding secret data within an ordinary, non-secret, file or message in order to avoid detection

Introduction

In cryptography, encryption is the process of encoding information. This process converts the original representation of the information, known as plaintext, into an alternative form known as ciphertext. Ideally, only authorized parties can decipher a ciphertext back to plaintext and access the original information.

The conversion of encrypted data into its original form is called Decryption. It is generally a reverse process of encryption. It decodes the encrypted information so that an authorized user can only decrypt the data because decryption requires a secret key or password.

Few common items that are encrypted include text files, images, e-mail messages, user data and directories. The recipient of decryption receives a prompt or window in which a password can be entered to access the encrypted data. It may also be performed with a set of keys or passwords.

The word Steganography is derived from two Greek words- 'stegos' meaning 'to cover' and 'grayfia', meaning 'writing', thus translating to 'covered writing', or 'hidden writing'. Steganography is a method of hiding secret data, by embedding it into an audio, video, image, or text file. It is one of the methods employed to protect secret or sensitive data from malicious attacks.

Cryptography and steganography are both methods used to hide or protect secret data. However, they differ in the respect that cryptography makes the data unreadable, or hides the meaning of the data, while steganography hides the existence of the data.

In this project we have used Caesar Cipher & Steganography . Which is the most used technique for secret messaging and exchanging of confidential data.

Problem Statement

- How to secure your information from unauthorized access?
- Secure message/ data transfer.
- Encoder-Decoder – Secures your Information by Encoding the Messages.

Hardware Requirements

SYSTEM :- INTEL CORE I3 (Minimum)
HARD DISK :- 512 GB (Minimum)
MONITOR :- STANDARD LED MONITOR
INPUT DEVICES :- KEYBOARD
RAM:- 4 GB AND ABOVE
PROCESSOR:- x32 and x64 bit.

Software Requirements

OPERATING SYSTEM :- WINDOWS 7 (Min)
PROGRAMMING LANGUAGE :- PYTHON
CODE EDITOR :- VS CODE / ATOM
LIBRARIES USED :- TKINTER & STEGANO
ENCRYPTING METHOD :- CAESAR CIPHER & STEGANOGRAPHY

Literature Survey

Technique	Caesar Cipher
Person who introduced	Julius Caesar
Year	Around 100 BC
Specification	One of the earliest and widely used method for encryption messages.
Cipher type	Substitution cipher, i.e., each letter of a given text is replaced by a letter some fixed number of positions down the alphabet.
Rotation Type	Mono-alphabetic Rotation
Example	If A is the 1st letter in message then with a shift of 1, A would be replaced by B, B would become C, and so on.
Representation	Modular arithmetic by first transforming the letters into numbers, according to the scheme, A = 0, B = 1,…, Z = 25.
Encryption Formula	$E_n(x) = (x + n) mod\ 26$
Breaking cipher	Ciphertext-only scenario.
Decryption Formula	$D_n(x) = (x - n) mod\ 26$
Use for	Securing data and Secure messaging

Technology used	Python, Tkinter, Stegano
Technique	Steganography
Person who introduced	Johannes Trithemius
Year	1499
Specification	One of the earliest and widely used method for hiding existence messages.
Example	It is known that during both world wars, female spies used knitting to send messages, perhaps making an irregular stitch or leaving an intentional hole in the fabric.
Breaking stegano	Same as that of encryption but reverse.
Representation	Every text is distributed into 3 pixels, and after every 3 pixel a binary number is added which hides the given text
Use for	Securing / hiding the confidential text or data existence.
Problem solved	Cyber attacks, Secure data transfer and also Securing confidential information

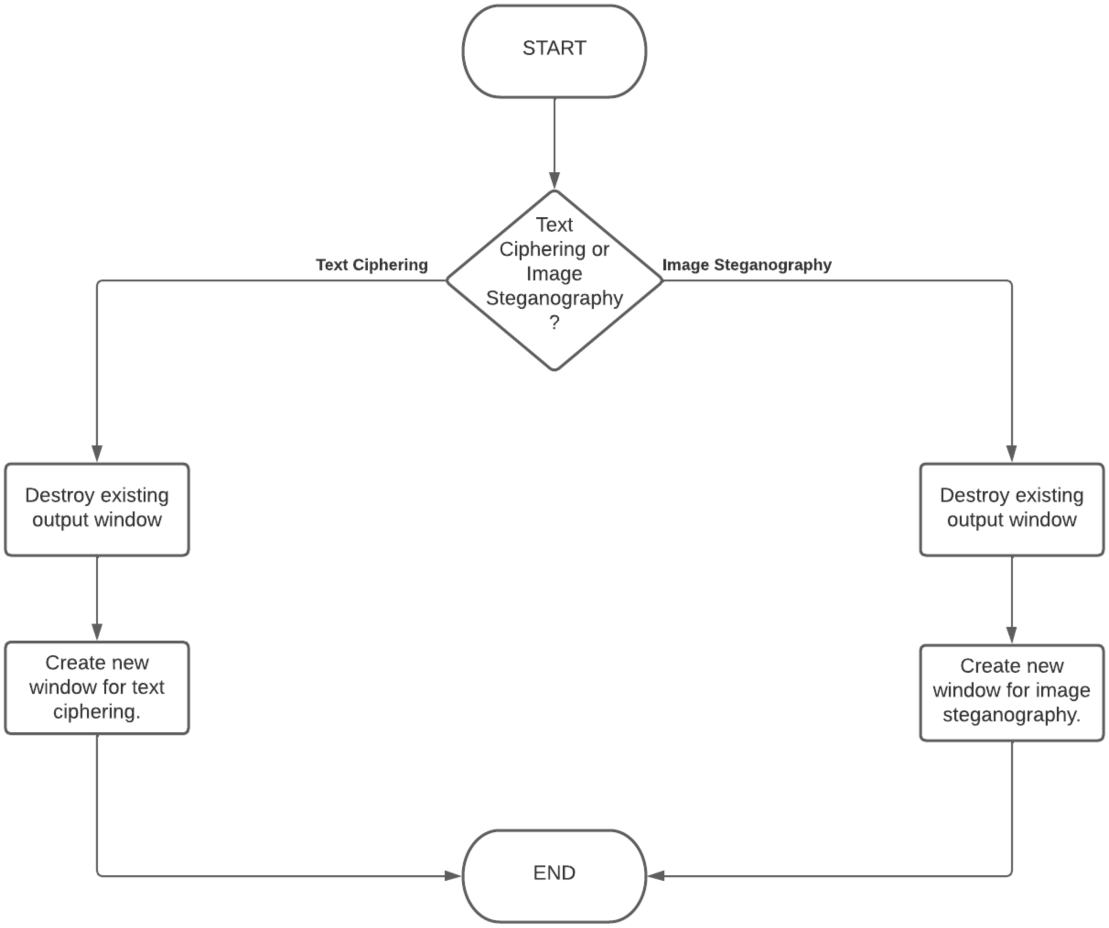
Methodology & Implementation For Caesar Cipher

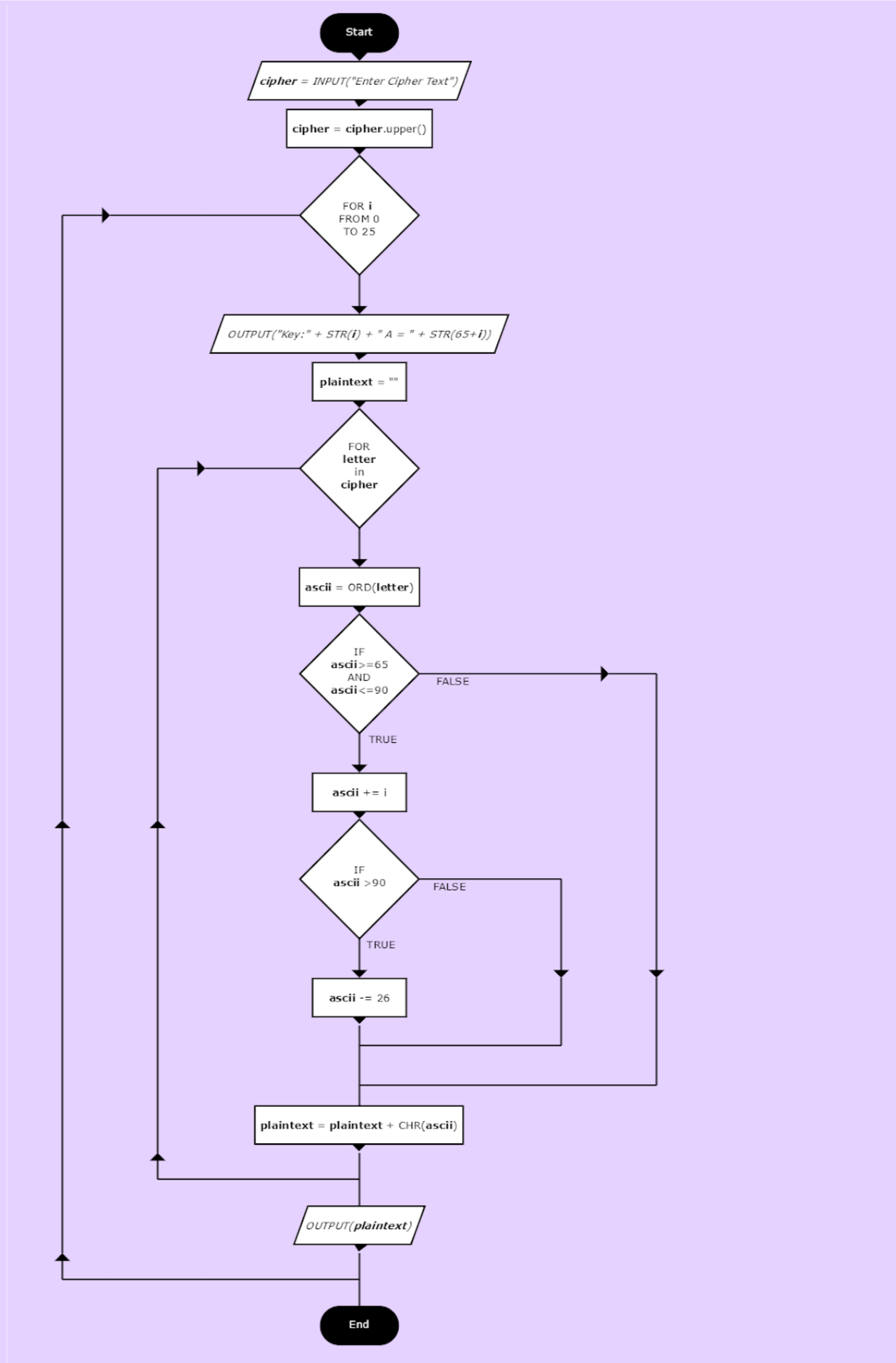
- Import tkinter, numpy libraries.
- Initialized window to cover the whole screen.
- Added labels, input fields and buttons.
- Created a function **encrypt()** to encode the input string entered by the user.
- Ask the user to provide a **key** which will help to decode the text while decrypting.
- Created a function **decrypt()** to decode the encrypted text.
- If the user enters the same key used while encrypting text, it will show the decoded text.
- Press the "**Show Result**" button to get the decoded text.

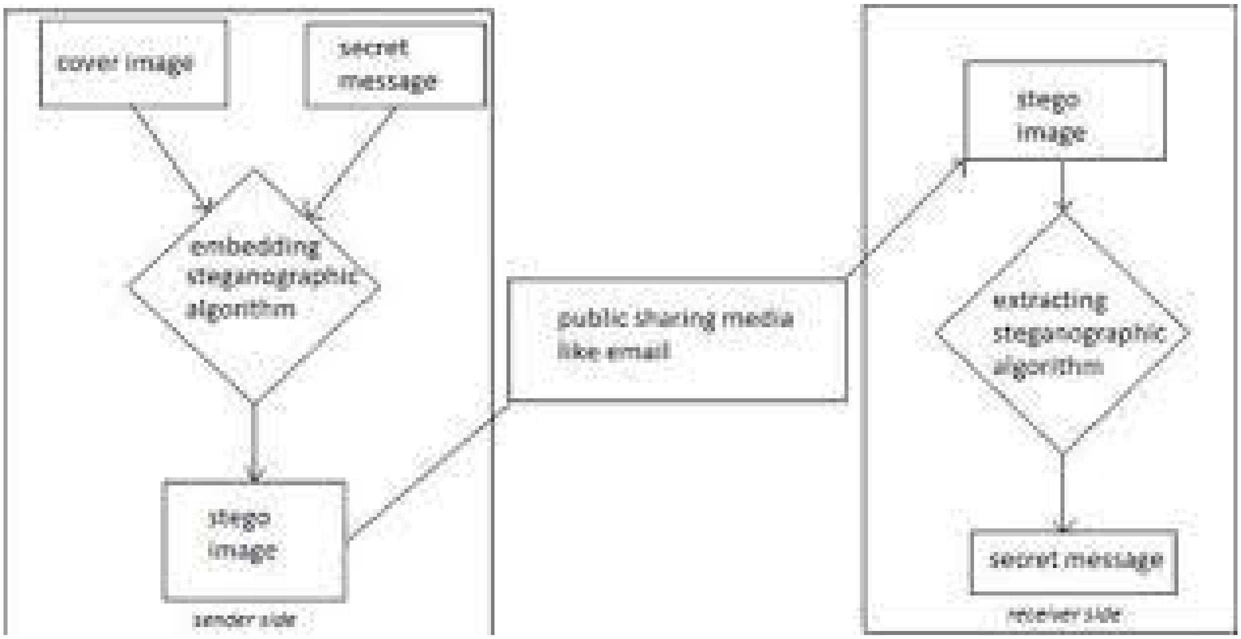
Methodology & Implementation For Steganography

- Import tkinter, stegano libraries.
- Initialized window to cover the whole screen.
- Added labels, input fields and buttons.
- Created a function **encode()** to encode the message and hide it into an image.
- A copy of that image will get downloaded in user's system.
- Created a function **decrypt()** to decode the encrypted image.
- Click "**Go**" button to get the encoded text.

Project Design


START
Text Ciphering or Image Steganography ?
Text Ciphering
Image Steganography
Destroy existing output window
Destroy existing output window
Create new window for text ciphering.
Create new window for image steganography.
END

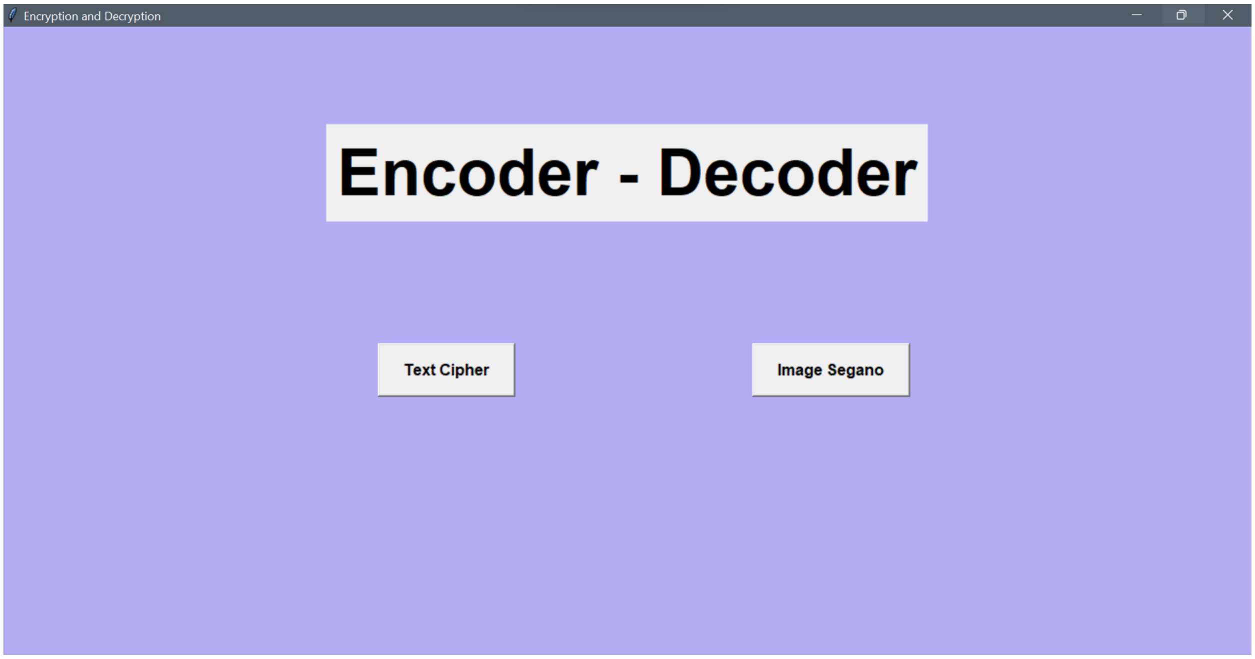

Start
cipher = INPUT("Enter Cipher Text")
cipher = cipher.upper()
FOR i FROM 0 TO 25
OUTPUT("Key:" + STR(i) + " A = " + STR(65+i))
plaintext = ""
FOR letter in cipher
ascii = ORD(letter)
IF ascii>=65 AND ascii<=90
FALSE
TRUE
ascii += i
IF ascii >90
FALSE
TRUE
ascii -= 26
plaintext = plaintext + CHR(ascii)
OUTPUT(plaintext)
End

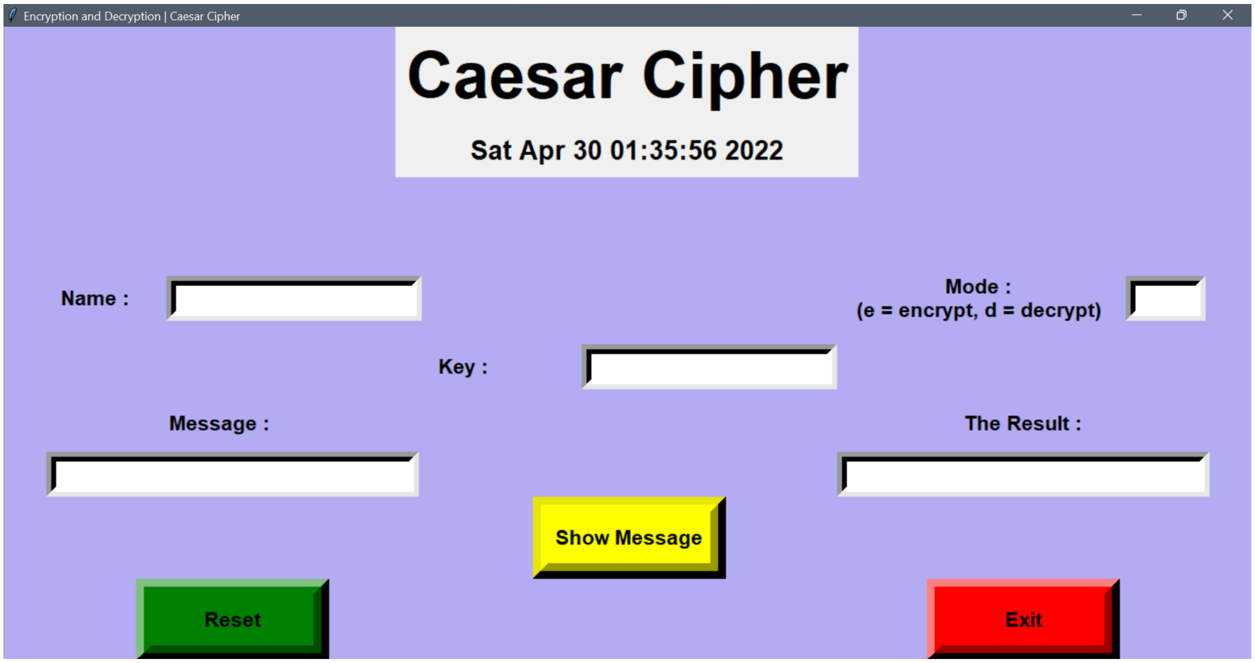

cover image
secret message
embedding steganographic algorithm
stego image
sender side
public sharing media like email
stego image
extracting steganographic algorithm
secret message
receiver side

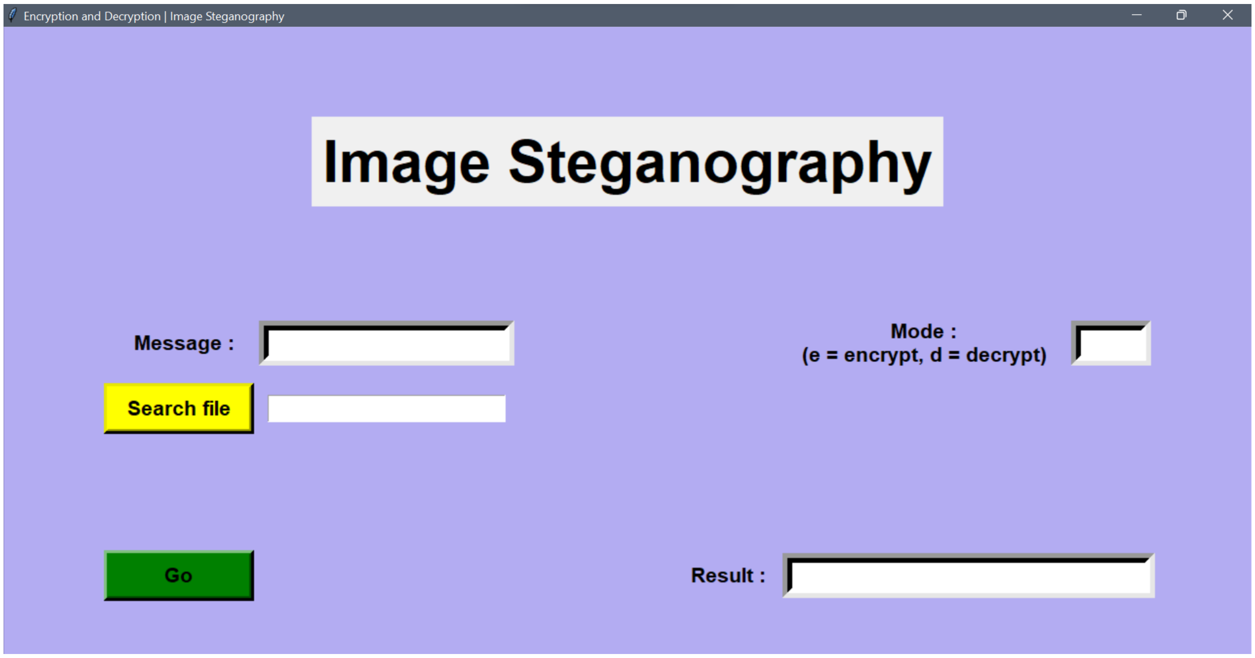
Project Code

```
from cProfile import label
from cgitb import text
from tkinter import *
# import other necessery modules
import random
import time
import datetime
from PIL import ImageTk,Image
from stegano import exifHeader as stg
from numpy import char, chararray
from tkinter import messagebox
from tkinter.filedialog import *
#from new import Encode

# creating root object
root = Tk()
# defining size of window
root.geometry("1920x1080")
# setting up the title of window
root.title("Encryption and Decryption")
root.config(bg='#b3acf2')

Tops = Frame(root, width = 2600, relief = SUNKEN, bg='#b3acf2')
Tops.pack(side = TOP)
lblInfo = Label(Tops, font = ('helvetica', 50, 'bold'), bg='#b3acf2', text = "", fg = "Black", bd = 10, anchor='w')
lblInfo.grid(row = 0, column = 0)
lblInfo = Label(Tops, font = ('helvetica', 50, 'bold'), text = "Encoder - Decoder", fg = "Black", bd = 10, anchor='w')
lblInfo.grid(row = 1, column = 0)

def windowForTextCipher():
    # Destroying optional window
    root.destroy()
    # Creating new window for text ciphering
    Screen = Tk()
    Screen.geometry("1920x1080")
    Screen.title("Encryption and Decryption | Caesar Cipher")
    Screen.config(bg='#b3acf2')

    # Creating table frame to contain rows and columns
    Tops = Frame(Screen, width = 2600, relief = SUNKEN)
    Tops.pack(side = TOP)
    f1 = Frame(Screen, width = 800, height = 700, relief = SUNKEN, bg='#b3acf2')
    f1.pack(side = BOTTOM)

    # ==============================================
    # TIME
    # ==============================================
    localtime = time.asctime(time.localtime(time.time()))
    lblInfo = Label(Tops, font = ('helvetica', 50, 'bold'), text = "Caesar Cipher", fg = "Black", bd = 10, anchor='w')
    lblInfo.grid(row = 0, column = 0)
    lblInfo = Label(Tops, font=('arial', 20, 'bold'), text = localtime, fg = "black", bd = 10, anchor = 'w')
    lblInfo.grid(row = 7, column = 0)
    rand = StringVar()
    Msg = StringVar()
    key = StringVar()
    mode = StringVar()
    Result = StringVar()
    # exit function
    def qExit():
        Screen.destroy()

    # Function to reset the window
    def Reset():
        rand.set("")
        Msg.set("")
        key.set("")
        mode.set("")
        Result.set("")

    # Column for Name
    lblReference = Label(f1, font = ('arial', 16, 'bold'), text = "Name :",width=7, bd = 16, anchor = "w")
    lblReference.grid(row = 0, column = 0)
    lblReference.configure(bg='#b3acf2')
    # Column for Name input
    txtReference = Entry(f1, font = ('arial', 16, 'bold'), textvariable = rand, bd = 10, insertwidth = 4, bg = "white", justify
= 'left')
    txtReference.grid(row = 0, column = 1)
    # Column for mode
    lblmode = Label(f1, font = ('arial', 16, 'bold'), text = "Mode :\n(e = encrypt, d = decrypt)", bd = 16, anchor = "w")
    lblmode.grid(row = 0, column = 4)
    lblmode.configure(bg='#b3acf2')
    # Column for mode input
    txtmode = Entry(f1, font = ('arial', 16, 'bold'), textvariable = mode, bd = 10, insertwidth = 4, width=5, bg = "white",
justify = 'center')
    txtmode.grid(row = 0, column = 5)
    #  Column for key
    lblkey = Label(f1, font = ('arial', 16, 'bold'), text = "Key :", width=10, bd = 16, anchor = "w")
    lblkey.grid(row = 1, column = 2)
    lblkey.configure(bg='#b3acf2')
    #  Column for key input
    txtkey = Entry(f1, font = ('arial', 16, 'bold'), textvariable = key, bd = 10 , insertwidth = 4, bg = "white", justify =
'center')
    txtkey.grid(row = 1, column = 3)
    #  Column for message
    lblMsg = Label(f1, font = ('arial', 16, 'bold'), text = "Message :", bd = 16,width=10, anchor = "w")
    lblMsg.grid(row = 2,columnspan=2, column = 0)
    lblMsg.configure(bg='#b3acf2')
    #  Column for message input
    txtMsg = Entry(f1, font = ('arial', 16, 'bold'), textvariable = Msg, bd = 10, insertwidth = 4, width=30, bg = "white",
justify = 'left')
    txtMsg.grid(row = 3, columnspan=2,column = 0)
    #  Column for result
    lblService = Label(f1, font = ('arial', 16, 'bold'), text = "The Result :", bd = 16, anchor = "w")
    lblService.grid(row = 2, columnspan=2, column = 4)
    lblService.configure(bg='#b3acf2')
    #  Column for result output
    txtService = Entry(f1, font = ('arial', 16, 'bold'), textvariable = Result, bd = 10, insertwidth = 4, width=30, bg =
"white", justify = 'left')
    txtService.grid(row = 3,columnspan=2, column = 4)
    # Function for encrypting message
    def encrypt(key, text):
        specialChar = """ !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~"""
        result = ""
        for i in range(len(text)):
            ch = text[i]
            # Encrypt uppercase characters
            if (ch.isupper()):
                result += chr((ord(ch) + key-65) % 26 + 65)
            #Numerical values
            elif (ord(ch) >= 48 and ord(ch) <= 57):
                result += chr((ord(ch) + key+10) % 10 + 48)
            # Encrypt special characters
            elif(ch in specialChar):
                newChar = (specialChar.index(ch)+key)%33
                result += specialChar[newChar]
            # Encrypt lowercase characters
            elif (ch.islower()):
                result += chr((ord(ch) + key-97) % 26 + 97)
        return result
    # Function for decrypting message
    def decrypt(key,text):
        specialChar = """ !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~"""
        result = ""
        # traverse text
        for i in range(len(text)):
            ch = text[i]
            # Decrypt uppercase characters
            if (ch.isupper()):
                result += chr((ord(ch) - key-65) % 26 + 65)
            # Decrypt Numerical  values
            elif (ord(ch) >= 48 and ord(ch) <= 57):
                result += chr(((ord(ch) - key+10) % 10 + 48)-6)
            # Decrypt special characters
            elif(ch in specialChar):
                newChar = (specialChar.index(ch)-key)%33
                result += specialChar[newChar]
            # Decrypt lowercase characters
            elif(ch.islower()):
                result += chr((ord(ch) - key-97) % 26 + 97)

        return result
    # Function to check whether to encrypt or decrypt text based on 'mode' input
    def Ref():
        clear = Msg.get()
        k = int(key.get())
        m = mode.get()
        if (m == 'e'):
            Result.set(encrypt(k, clear))
        else:
            Result.set(decrypt(k, clear))
    # Reset button
    btnReset = Button(f1, padx = 16, pady = 8, bd = 16,
    fg = "black", font = ('arial', 16, 'bold'),
    width = 10, text = "Reset", bg = "green",
    command = Reset).grid(row = 8,  columnspan=2, column = 0)

    # Show message button
    btnTotal = Button(f1, padx = 16, pady = 8, bd = 16, fg = "black",
    font = ('arial', 16, 'bold'), width = 10,
    text = "Show Message", bg = "yellow",
    command = Ref).grid(row = 7, columnspan=2, column = 2)

    # Exit button
    btnExit = Button(f1, padx = 16, pady = 8, bd = 16,
    fg = "black", font = ('arial', 16, 'bold'),
    width = 10, text = "Exit", bg = "red",
    command = qExit).grid(row = 8,  columnspan=2, column = 4)


def windowForImageCipher():
    # Deleting optional window
    root.destroy()
    # Creating new window
    Screen = Tk()
    Screen.geometry("1920x1080")
    Screen.title("Encryption and Decryption | Image Steganography")
    Screen.config(bg='#b3acf2')

    # Creating frames
    Tops = Frame(Screen, width = 2600, relief = SUNKEN,  bg='#b3acf2')
    Tops.pack(side = TOP)
    f1 = Frame(Screen, width = 800, height = 700, relief = SUNKEN, bg='#b3acf2')
    f1.pack(side = BOTTOM)
    Msg = StringVar()
    addr = StringVar()
    mode = StringVar()
    Result = StringVar()

    # Function to check whether to encrypt or decrypt text based on 'mode' input
    def Ref():
        userMessage = Msg.get()
        fileAddr = addr.get()
        m = mode.get()
        if (m == 'e'):
            Encode(userMessage, fileAddr)
        else:
            Result.set(Decode(fileAddr))

    #  Function to browse file and store it's into a variable as string
    def OpenFile():
        global FileOpen
        FileOpen=StringVar()
        FileOpen = askopenfilename(initialdir ="/Desktop",title="SelectFile",filetypes=(("only jpeg files","jpg"),("all type
of files",".*")))
        addr.set(FileOpen)

    # Function to encode text into image
    def Encode(userMessge, fileAddr):
        Response = messagebox.askyesno("PopUp","Do you want to encode the image?")
        if Response == 1:
            stg.hide(FileOpen,fileAddr+".jpg",userMessge)
            messagebox.showinfo("Pop Up","Successfully Encoded")
        else:
            messagebox.showwarning("Pop Up","Unsuccessful, please try again")

    # Function to decode image into text
    def Decode(fileAddr):
        Message=stg.reveal(fileAddr)
        return Message

    # Column for Heading
    lblInfo = Label(Tops, font = ('helvetica', 45, 'bold'), bg='#b3acf2', text = "", fg = "Black", bd = 10, anchor='w')
    lblInfo.grid(row = 0, column = 0)
    lblInfo = Label(Tops, font = ('helvetica', 45, 'bold'), text = "Image Steganography", fg = "Black", bd = 10,
anchor='w')
    lblInfo.grid(row = 1, column = 0)

    # Column for Message
    lblReference = Label(f1, font = ('arial', 16, 'bold'), text = "Message :", width=7, bd = 16, anchor = "w")
    lblReference.grid(row = 0, column = 0)
    lblReference.configure(bg='#b3acf2')
    # Column for Message Input
    txtReference = Entry(f1, font = ('arial', 16, 'bold'), textvariable = Msg, bd = 10, insertwidth = 4, bg = "white", justify =
'left')
    txtReference.grid(row = 0, column = 1)

    # Column for Mode
    lblmode = Label(f1, font = ('arial', 16, 'bold'), text = "Mode :\n(e = encrypt, d = decrypt)", bd = 16, anchor = "w")
    lblmode.grid(row = 0, column = 5)
    lblmode.configure(bg='#b3acf2')
    # Column for Mode Input
    txtmode = Entry(f1, font = ('arial', 16, 'bold'), textvariable = mode, bd = 10, insertwidth = 4, width=5, bg = "white",
justify = 'center')
    txtmode.grid(row = 0, column = 6)
    searchFile = Button(f1, padx = 5, pady = 3, bd = 5, fg = "black",
    font = ('arial', 16, 'bold'), width = 10,
    text = "Search file", bg = "yellow",
    command = OpenFile).grid(row = 1, column = 0)
    # 2 Empty columns for blank spaces
    emptylabel = Label(f1, font = ('arial', 16, 'bold'), text = "",width=10,  bd = 16, anchor = "w")
    emptylabel.grid(row = 0, column = 3)
    emptylabel.configure(bg='#b3acf2')
    emptylabel1 = Label(f1, font = ('arial', 16, 'bold'), text = "", bd = 16, anchor = "w")
    emptylabel1.grid(row = 0, column = 4)
    emptylabel1.configure(bg='#b3acf2')
    # Column for file name display
    addrReference = Entry(f1, font = ('arial', 16, 'bold'), textvariable = addr, justify = 'left')
    addrReference.grid(row = 1, column = 1)
    # Empty column for blank spaces
    emptylabel2 = Label(f1, font = ('arial', 16, 'bold'), text = "", bd = 16, anchor = "w")
    emptylabel2.grid(row = 2, column = 0)
    emptylabel2.configure(bg='#b3acf2')
    emptylabel3 = Label(f1, font = ('arial', 16, 'bold'), text = "",width=10,  bd = 16, anchor = "w")
    emptylabel3.grid(row = 3, column = 0)
    emptylabel3.configure(bg='#b3acf2')
    processIntr = Button(f1, padx = 5, pady = 3, bd = 5, fg = "black",
    font = ('arial', 16, 'bold'), width = 10,
    text = "Go", bg = "green",
    command = Ref).grid(row = 4, column = 0)
    # Column for show result
    showResult = Label(f1, font = ('arial', 16, 'bold'), text = "Result :", bd = 16, anchor = "w")
    showResult.grid(row = 4,  column = 4)
    showResult.configure(bg='#b3acf2')
    # Column for show result output
    resultMessage = Entry(f1, font = ('arial', 16, 'bold'), textvariable = Result, bd = 10, insertwidth = 4, width=30, bg =
"white", justify = 'left')
    resultMessage.grid(row = 4, columnspan= 2, column = 5)
    # Empty column for blank spaces
    emptylabel2 = Label(f1, font = ('arial', 16, 'bold'), text = "", bd = 16, anchor = "w")
    emptylabel2.grid(row = 5, column = 0)
    emptylabel2.configure(bg='#b3acf2')


textCiphering = Button(text="Text Cipher",padx = 20, pady = 12, font = ('arial', 12, 'bold'),
command=windowForTextCipher)
textCiphering.place(relx=0.3,rely=0.5)

textCiphering = Button(text="Image Segano",padx = 20, pady = 12,font = ('arial', 12, 'bold'),
command=windowForImageCipher)
textCiphering.place(relx=0.6,rely=0.5)

root.mainloop()
```

Project Result






Encryption and Decryption | Image Steganography
Image Steganography
Message :
Mode :
(e = encrypt, d = decrypt)
Search file
Go
Result :

Applications

1) **Encryption/Decryption in email:** Email encryption is a method of securing the content of emails from anyone outside of the email conversation looking to obtain a participant's information. In its encrypted form, an email is no longer readable by a human. Only with your private email key can your emails be unlocked and decrypted back into the original message.

2) **Defense Government Organizations :** to facilitate secret communication

3) **For sending highly confidential messages or details on Social Media like Card details or Bank Information.**

4) **Encryption is also used to protect data in transit**, for example data being transferred via networks (e.g. the Internet, e-commerce), mobile telephones, wireless microphones, wireless intercom systems, Bluetooth devices and bank automatic teller machines. There have been numerous reports of data in transit being intercepted in recent years.

5) **Encryption can be used to protect data "at rest"**, such as information stored on computers and storage devices (e.g. USB flash drives). In recent years, there have been numerous reports of confidential data, such as customers' personal records, being exposed through loss or theft of laptops or backup drives; encrypting such files at rest helps protect them if physical security measures fail.

6) **Digital rights management systems**, which prevent unauthorized use or reproduction of copyrighted material and protect software against reverse engineering (see also copy protection), is another somewhat different example of using encryption on data at rest.

Conclusion

Early encryption techniques were often utilized in military messaging. Since then, new techniques have emerged and become common place in all areas of modern computing.
In today's world as Cyber Attacks have grown in large number there is a need to secure our data.

Thus, we have successfully developed an Encoder-Decoder project in Python. We used the popular tkinter library for rendering graphics on a display window and encoded - decoded using the Caesar Cipher method and Steganography .
In this way, we can encode our message, images and decode the encoded message, image in a secure way by using the key.

Future Scope

1. More encoding cipher options could be added such as :
   Advanced Encryption Standard (AES),
   Triple DES (Data Encryption Standard).

2. More secure and user oriented encryption can be done

3. It will be used in all purpose such as Internet banking, Sharing Personal details, Military & Defence connections and also identifing Terrorist threats , Securing your data in own devices more safely.

Reference

1) Tkinter library:
https://docs.python.org/3/library/tkinter.html
https://www.tutorialspoint.com/python/python_gui_programming.htm#%20:~:text=Tkinter%20is%20the%20standard%20GUI,to%20the%20Tk%%2020GUI%20toolkit.&text=Import%20the%20Tkinter%20module

2)Cesar Cipher:
https://en.wikipedia.org/wiki/Caesar_cipher
https://cryptography.fandom.com/wiki/Caesar_cipher

3)Steganography:
https://en.wikipedia.org/wiki/Steganography
https://www.techtarget.com/searchsecurity/definition/steganography