Mini Project

“CAESAR CIPHER”

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 "**CAESAR CIPHER**" 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 "CAESAR CIPHER" 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 . This project will be built using Python. 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. This processes are called the Encryption-Decryption processes. In this project, users have to enter the message whether to encode or decode. Users have to select the mode to choose the encoding and decoding process. The same key must be used to process the encoding and decoding for the same message. To build this project we will use the basic concept of python, Tkinter.

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. One of the reasons for implementing an encryption-decryption system is privacy. As information travels over the Internet, it is necessary to scrutinize the access from unauthorized organizations or individuals. Due to this, the data is encrypted to reduce data loss and theft. 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. For decryption, the system extracts and converts the garbled data and transforms it into words and images that are easily understandable not only by a reader but also by a system. Decryption can be done manually or automatically. It may also be performed with a set of keys or passwords. Here we have used Caesar Cipher. Which is one of 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
PROCESSOR:- x32 and x64 bit.

Software Requirements

OPERATING SYSTEM :- WINDOWS 7 (Min)
PROGRAMMING LANGUAGE :- PYTHON
CODE EDITOR :- VS CODE / ATOM
LIBRARIES USED :- TKINTER & BASE 64
ENCRYPTING METHOD :- CAESAR CIPHER

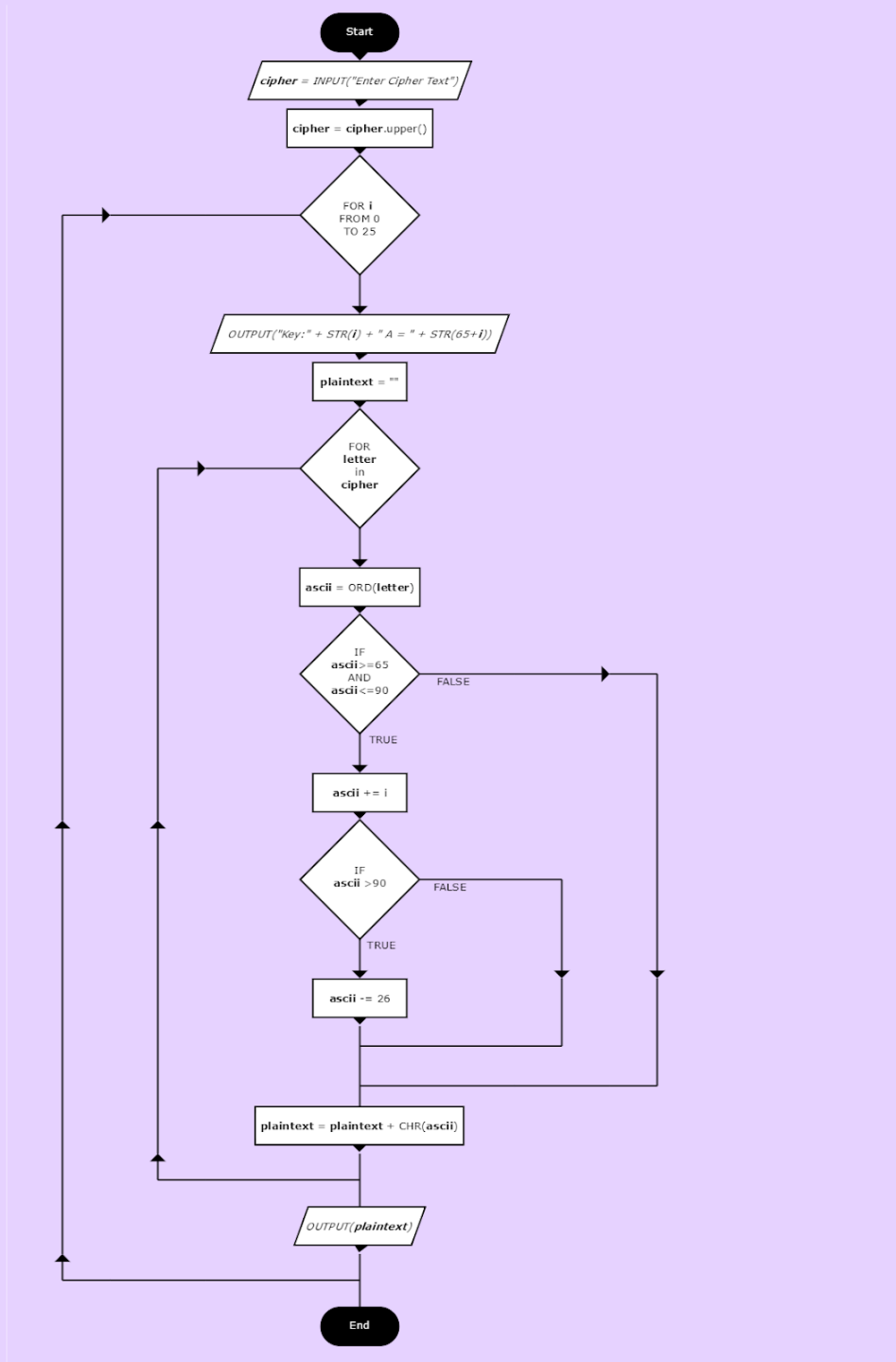
Literature Survey

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	$\boxed{E_n(x) = (x + n) mod\ 26}$
Breaking cipher	Ciphertext-only scenario.
Decryption Formula	$D_n(x) = (x - n) mod\ 26$
Use for	Securing data and Secure messaging
Technology used	Python,Tintker

Methodology & Implementation

- 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.

Project Design



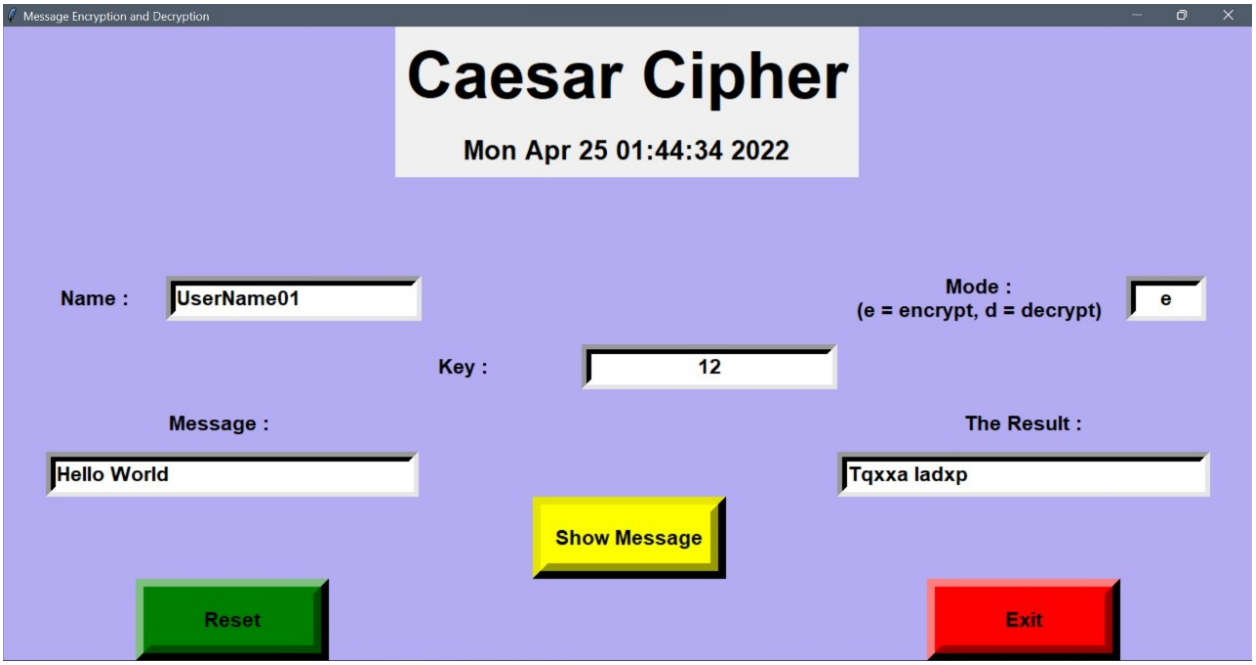
Project Code

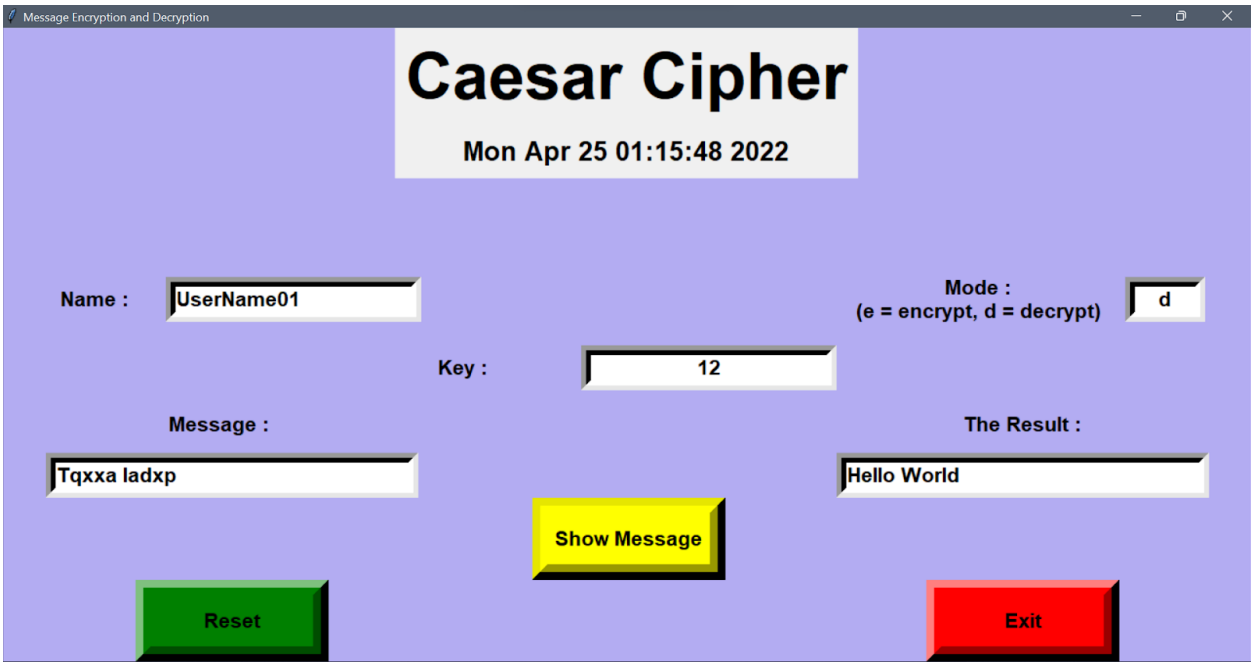
```
from tkinter import *
import re
# import other necessery modules
import random
import time
import datetime

from numpy import char, chararray
# creating root object
root = Tk()
# defining size of window
root.geometry("1920x1080")
# setting up the title of window
root.title("Message Encryption and Decryption")
root.configure(bg='#b3acf2')
Tops = Frame(root, width = 2600, relief = SUNKEN)
Tops.pack(side = TOP)
f1 = Frame(root, 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():
    root.destroy()
# Function to reset the window
def Reset():
    rand.set("")
    Msg.set("")
    key.set("")
    mode.set("")
    Result.set("")
# reference
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')
txtReference = Entry(f1, font = ('arial', 16, 'bold'), textvariable = rand, bd = 10, insertwidth = 4, bg = "white", justify =
'left')
txtReference.grid(row = 0, column = 1)
# labels
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')
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)

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')
txtkey = Entry(f1, font = ('arial', 16, 'bold'), textvariable = key, bd = 10 , insertwidth = 4, bg = "white", justify = 'center')
txtkey.grid(row = 1, column = 3)

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')
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)

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')
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)


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


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


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)


# keeps window alive
root.mainloop()
```

Project 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. In this way, we can encode our message and decode the encoded message in a secure way by using the key.

Future Scope

1. More encoding cipher options could be added such as :
   Steganography,
   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://www.geeksforgeeks.org/caesar-cipher-in-cryptography/

https://cryptography.fandom.com/wiki/Caesar_cipher