

SOFTWARE ENGINEERING

MESSAGE ENCRYPTER AND DECRYPTER

Group Members

<u>Name</u>	<u>Roll Number</u>
Ritik Raj Kumar	16500119022
Kamal Raj	16500119031
Aditya Kumar	16500119043
Ashish Ranjan	16500119039
Gufran Hasan	16500119034
Shadab Khan	16500119025

Faculty Name

ABH

Associate Professor

(CSE, CIEM, Kolkata)



Abstract

Nowadays, the use of devices such as computers, mobile and many more other devices for communication as well as for data storage and transmission has increased. As a result, there is an increase in the number of users. Along with these users, there is also an increase in the number of unauthorized users who are always trying to access data by unfair means. This arises the concern of data security. Messages are sent over an insecure transmission channel from different sources, some messages themselves are highly confidential hence, securing them from any attack is essentially required.

To solve this problem, we are using the Vigenère cipher algorithm for encrypting and decrypting the messages. This encrypted message is unrecognizable to unauthorized users, and it can be sent over any network and can be decrypted by using only the same algorithm at the receiving end. Hence, it ensures secure transmission of the message.

Aim of the project

The project aims to develop a secure way of transferring messages between the sender and the receiver. The message should be encrypted before it is being sent over a network and it should be correctly decrypted on the receiver side.

Objective of the project

The objective of this project is to encrypt and decrypt messages using a common password/key. This project will be built using the *Tkinter* library and the *base64* library. We have also used another library named *Auto PY to EXE* to convert the program into an executable file so that it will become portable and much easier for the users to use.

In this project, users will have to enter the message to either encrypt or decrypt and select the mode to choose for the encrypting and decrypting process. The same password/key must be used to process the encrypting and decrypting process for the same message.

Scope of the project

Product scope

The project works by encrypting the given message by using the Vigenère cipher algorithm so that this message can be sent securely over any network. On the receiver side, the receiver has a program for decrypting the ciphertext to the original message. This helps in sending confidential and sensitive information/messages securely over the internet. The main application of this can be very helpful in medical and military fields.

Design and Implementation constraints

- i) Python must be used for the front-end.
- ii) Encryption and Decryption should be done using the Vigenère cipher algorithm.
- iii) The original message must be in .txt format.

Assumptions and Dependencies

- i) The sender and the receiver are connected over a network.

Functional Requirements

- i) The system shall encrypt the given message to a meaningless text called ciphertext.
- ii) The system shall decrypt the received encrypted message to an understandable message. This is done using the Vigenère cipher algorithm.
- iii) The system ensures that the message is securely sent over any transmission medium so that a third-party system cannot make modifications to the message being sent as unauthorized access is not supported.

Non-Functional Requirements

Performance Requirements

- i) For smooth & efficient encryption, the message should be in .txt format.
- ii) Decryption should not take more than a few seconds.

Safety and Security Requirements

- i) If the decryption takes more than a few seconds, then discard the message (as the message might have been corrupted during the transmission) and ask the sender to re-send it.
- ii) Encryption is done using an encryption key. Decryption will happen only when the same encryption key is used on the receiver side.

Software Quality Attributes

Reliability

External factors do not affect the system. Vigenère cipher algorithm is universally accepted and generates consistent results therefore there are very few chances of errors. An error can occur only if there is a transmission glitch (the probability of which is very rare). So, the system is reliable.

Usability

It uses a python-based GUI which is very user-friendly and provides buttons for easy navigation. A person with a basic understanding of computers can easily use this software for encrypting/decrypting the message using a password/key.

Testability

The system is easy to test and find defects. The system is divided into different modules performing specific functions that can be tested individually.

External Interface Requirements

User Interface

The interface window contains two text boxes, one for entering the message and the other one for entering the password/key. Along with this box, it also contains two radio buttons (*Encrypt/Decrypt*) for selecting the modes. After selecting the mode, click on the button *View Result* to get the Encrypted/Decrypted text. It also contains some other buttons for different functions, and they are as follows:

- i) *Save Result*: To save the result in a .txt file.
- ii) *Select File*: To load the contents of a text file into the *Enter Your Text* textbox.
- iii) *Reset*: To reset the program.
- iv) *Exit*: To exit the program.

Hardware Interfaces

- i) Sender Computer: For encrypting the original message.
- ii) Receiver Computer: For decrypting the encrypted message.

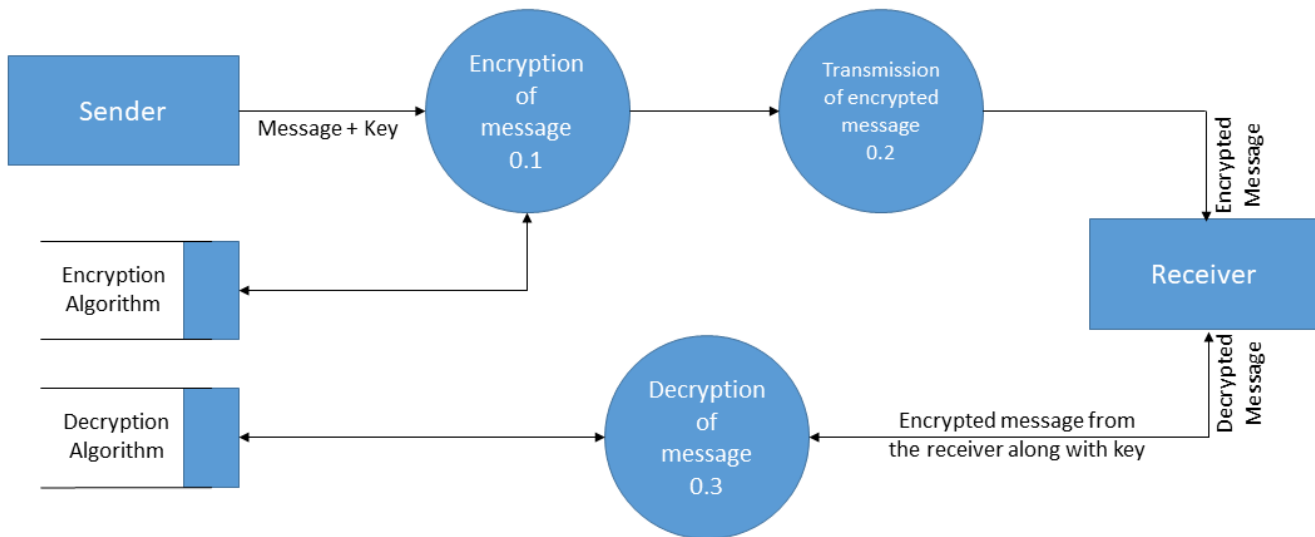
Software Interfaces

The frontend is made using a python module named *Tkinter()*. This is a standard GUI python library, and it provides an interactive window for the user. The user needs to enter the message that has to be encrypted/decrypted. After this, the user will have to provide the password/key and then select the mode (*Encrypt/Decrypt*) for encrypting or decrypting the message. After these details are entered, we use the python functions declared in the program to encrypt/decrypt the message using the Vigenère cipher algorithm. When this process is done, the user will get the encrypted/decrypted message as output.

Dataflow Diagram

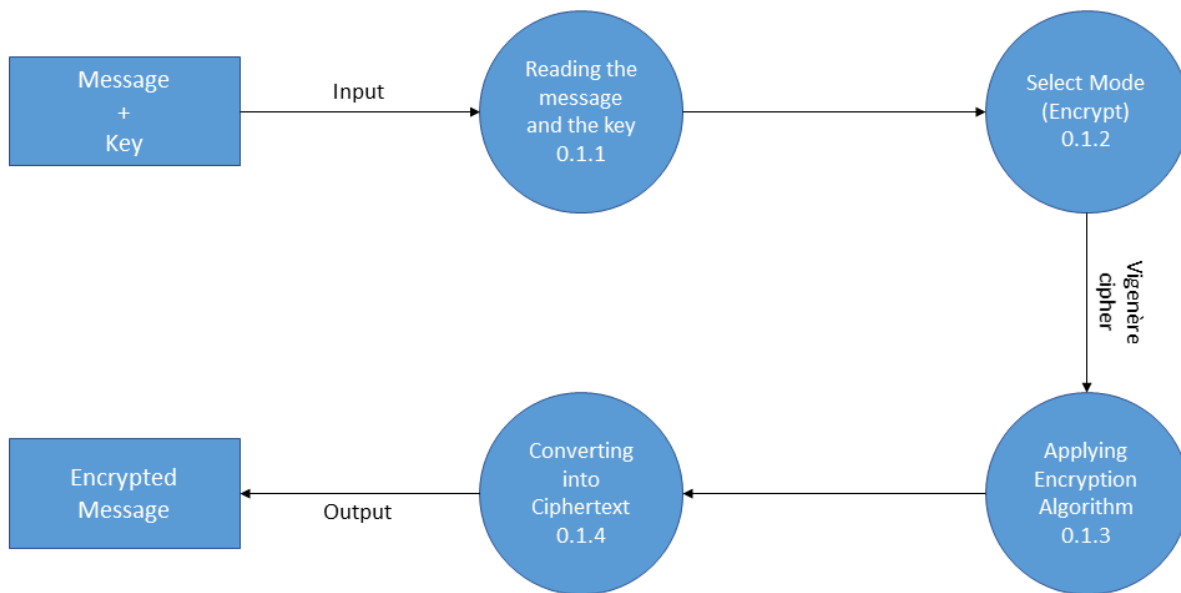


(a) Level 0 DFD (Context Diagram)

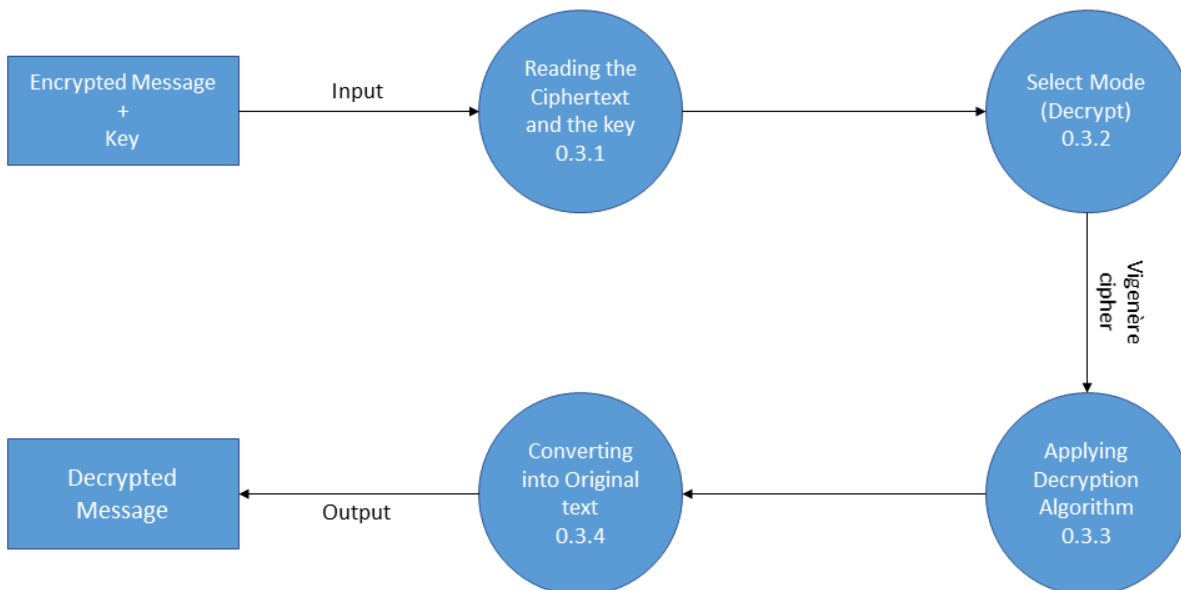


(b) Level 1 DFD

Encryption of Message



Decryption of Message



(c) Level 2 DFD

Implementation

Source Code

```
#import required modules
import base64
from tkinter import *
from tkinter import messagebox as mb
from tkinter.filedialog import askopenfilename, asksaveasfile

#initialize window
root = Tk()
root.geometry('600x400')
root.configure(bg = 'dimgray')
root.resizable(0, 0)

#title of the window
root.title("Message Encrypter and Decrypter")

#window labels
Label(root, text = 'Message Encrypter and Decrypter', font = 'arial 20 bold', bg = 'dimgray', fg = 'ghostwhite').pack()
Label(root, text = 'Developed by - \n Ritik Raj Kumar, Kamal Raj, Aditya Kumar, Ashish Ranjan, Gufran Hasan, Md. Shadab Khan', font = 'arial 8 bold', bg = 'dimgray', fg = 'ghostwhite').pack(side = BOTTOM)

#define variables
Text = StringVar()
private_key = StringVar()
mode = IntVar()
Result = StringVar()
textfile = StringVar()

"""Define Functions"""

#function to encode
def Encode(key, message):
    enc = []
    for i in range(len(message)):
        key_c = key[i % len(key)]
        enc.append(chr((ord(message[i]) + ord(key_c)) % 256))

    return base64.urlsafe_b64encode(''.join(enc).encode()).decode()

#function to decode
```

```
def Decode(key, message):
    dec = []
    message = base64.urlsafe_b64decode(message).decode()
    for i in range(len(message)):
        key_c = key[i % len(key)]
        dec.append(chr((256 + ord(message[i]) - ord(key_c)) % 256))

    return "".join(dec)
```

#function to set mode

```
def Mode():
    while True:
        if Text.get() == "":
            mb.showwarning('Warning', 'Enter Your Text!')
            break

        elif private_key.get() == "":
            mb.showwarning('Error', 'Enter Password')
            break

        if mode.get() == 0:
            mb.showwarning('Error!', 'Select Mode')
            break

        elif mode.get() == 1:
            Result.set(Encode(private_key.get(), Text.get()))
            mb.showinfo('Success', 'Encryption Successful!')
            break

        elif mode.get() == 2:
            Result.set(Decode(private_key.get(), Text.get()))
            mb.showinfo('Success', 'Decryption Successful!')
            break
```

#function to exit window

```
def Exit():
    root.destroy()
```

#function to reset

```
def Reset():
    Text.set("")
    private_key.set("")
    mode.set("0")
    Result.set("")
```



```

textfile.set("")

#function to load a text file file
def choosefile():

    #shows an "Open" dialog box and return the path to the selected file
    file1 = askopenfilename(title = "Select a text file", filetypes = (("text files", "*.txt"),))
    file1 = open(file1)
    data = file1.read()
    Text.set(data)

#function to save the result in .txt format
def savefile():
    while True:
        if Result.get() == "":
            mb.showinfo('Warning!', 'First Encrypt or Decrypt Text1')
            break
        tf = asksaveasfile(
            mode = 'w', title = "Save Result", defaulttextension = ".txt")
        tf.write(f"{Result.get()}")
        break

"""Labels and Buttons"""

#message
Label(root, font = 'arial 12 bold', text = 'Enter Your Text :', bg = 'dimgray', fg = 'ghostwhite').place(x = 50, y = 60)
file3 = Entry(root, font = 'arial 12 ', textvariable = Text, width= '38', bg = 'dimgray', fg = 'ghostwhite').place(x = 200, y = 60)

#key/password
Label(root, font = 'arial 12 bold', text = 'Enter Password :', bg = 'dimgray', fg = 'ghostwhite').place(x = 50, y = 100)
Entry(root, font = 'arial 12', textvariable = private_key , width= '38', bg = 'dimgray', fg = 'ghostwhite').place(x = 200, y = 100)

#mode
Label(root, font = 'arial 12 bold', text = 'Select Mode :', bg = 'dimgray', fg = 'ghostwhite').place(x = 50, y = 140)
Radiobutton(root, font = 'arial 10 bold', text = "Encrypt", bg = 'dimgray', variable = mode, value = 1).place(x = 200, y = 140)
Radiobutton(root, font = 'arial 10 bold', text = "Decrypt", bg = 'dimgray', variable = mode, value = 2).place(x = 300, y = 140)

```

```
#result button
Button(root, font = 'arial 10 bold', text = 'View Result', padx = 2, bg = 'Cyan', command = Mode).place(x = 255,
y = 180)

#save file button
Button(root, font = 'arial 10 bold', text = 'Save Result', padx = 2, bg = 'ghostwhite', command =
savefile).place(x = 110, y = 220)

#choose file button
Button(root, font = 'arial 10 bold', text = 'Select File', padx = 2, bg = 'ghostwhite', command =
choosefile).place(x = 400, y = 220)

#Encrypted/Decrypted Text
Label(root, font = 'arial 12 bold', text = 'Encrypted/Decrypted Text :', bg = 'dimgray', fg = 'ghostwhite').place(x
= 200, y = 260)
Entry(root, font = 'arial 12', textvariable = Result, width= '55', bg = 'dimgray', fg = 'ghostwhite').place(x = 50, y
= 290)

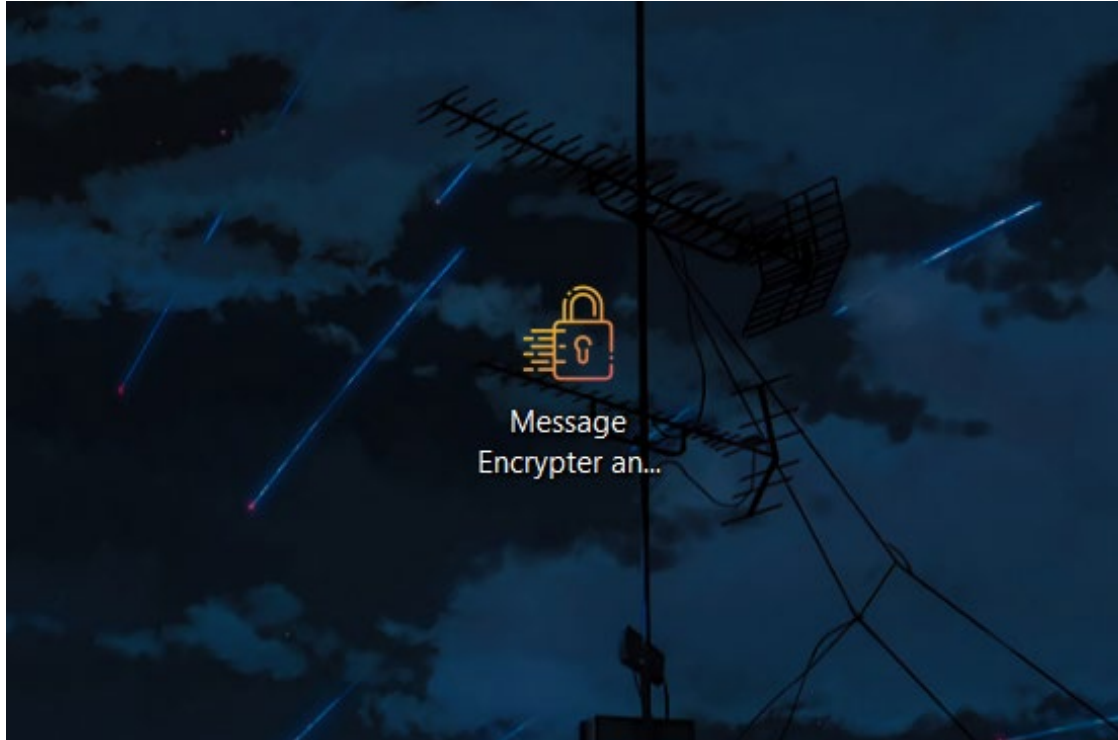
#reset button
Button(root, font = 'arial 10 bold', text = 'Reset', width = 6, command = Reset, bg = 'Green', padx=2).place(x =
120, y = 340)

#exit button
Button(root, font = 'arial 10 bold', text= 'Exit' , width = 6, command = Exit, bg = 'Red', padx = 2, pady =
2).place(x = 410, y = 340)

#main
root.mainloop()
```

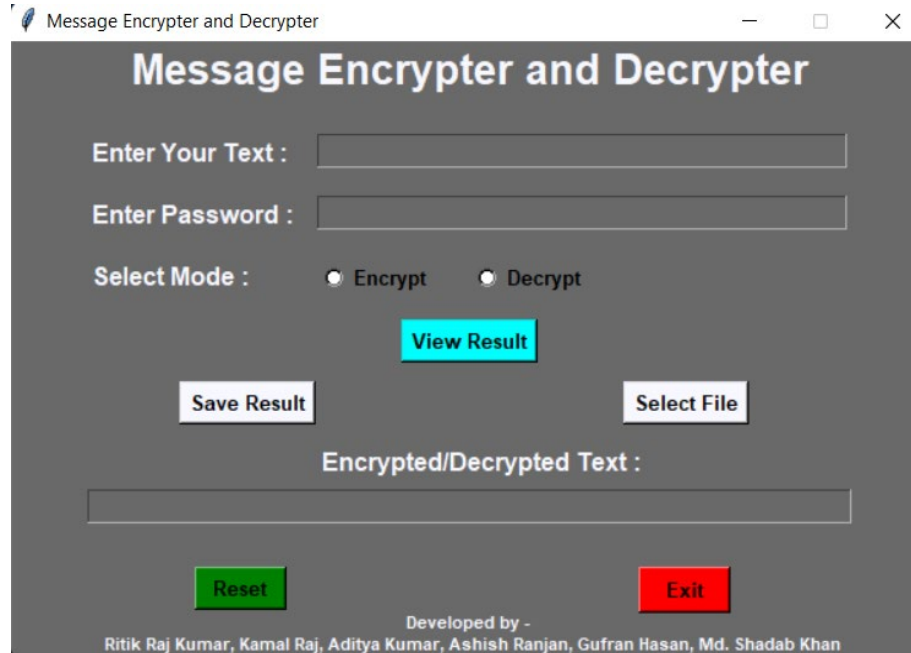
Screenshots of Implementations

Executable File



For the Encryption process

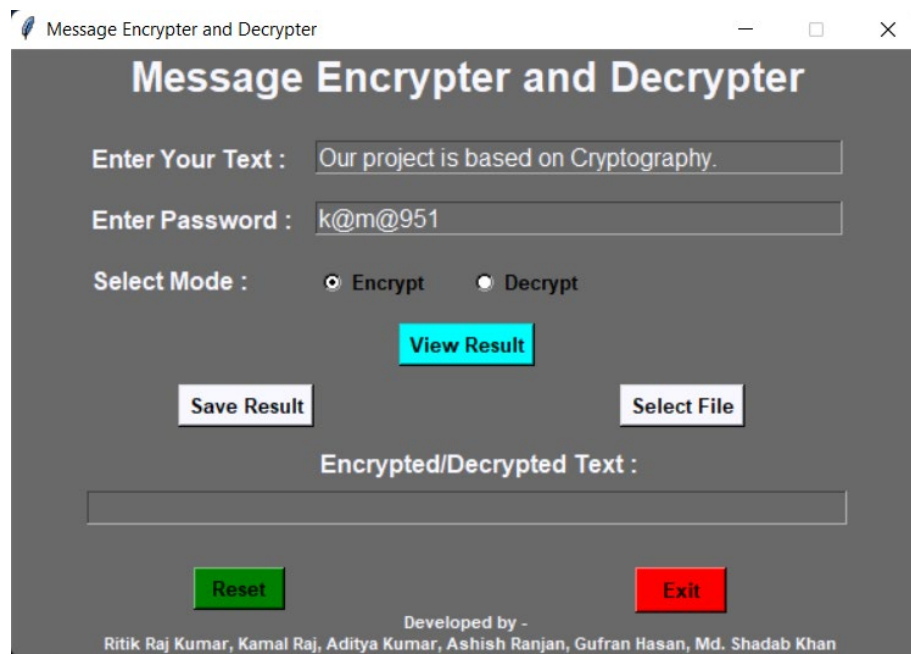
Application first screen asking for message, password, and mode:



The screenshot shows a window titled "Message Encrypter and Decrypter". The interface includes the following elements:

- Enter Your Text :** A text input field.
- Enter Password :** A password input field.
- Select Mode :** Two radio buttons labeled "Encrypt" and "Decrypt".
- View Result**: A cyan button.
- Save Result**: A white button.
- Select File**: A white button.
- Encrypted/Decrypted Text :** A large text area for the output.
- Reset**: A green button.
- Exit**: A red button.
- Footer**: Text stating "Developed by - Ritik Raj Kumar, Kamal Raj, Aditya Kumar, Ashish Ranjan, Gufran Hasan, Md. Shadab Khan".

After entering message and password, and selecting mode (Encrypt):



The screenshot shows the same application window after user input. The "Encrypt" mode is selected. The input fields are populated with the following text:

- Enter Your Text :** "Our project is based on Cryptography."
- Enter Password :** "k@m@951"

The "View Result" button is highlighted in cyan, indicating it is the active or next step. The "Save Result" and "Select File" buttons remain white. The "Encrypted/Decrypted Text" area is empty. The "Reset" and "Exit" buttons are green and red respectively. The footer text remains the same.

After clicking **View Result** button:

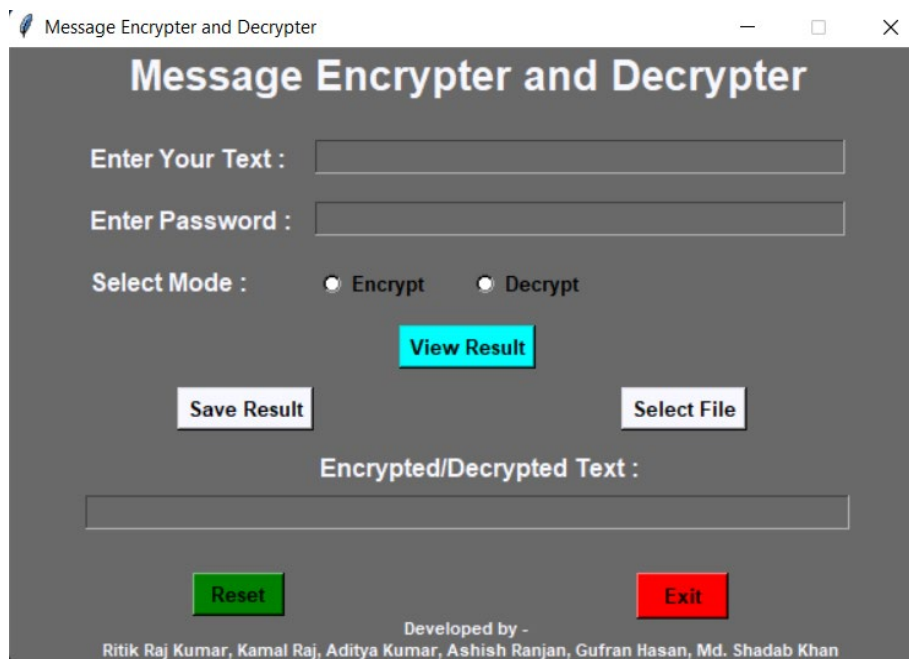


Encrypted Message:



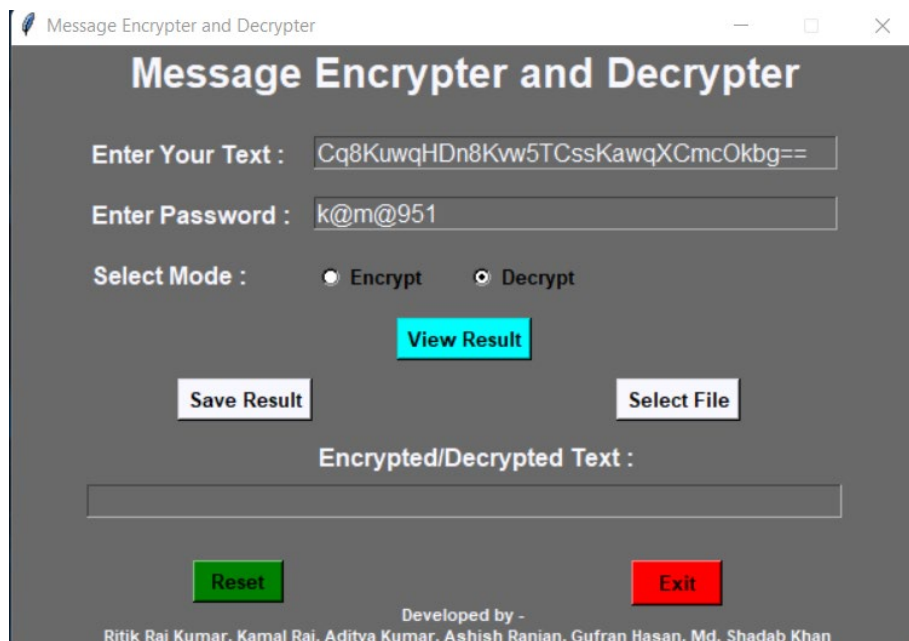
For the Decryption process

Application first screen asking for message, password, and mode:



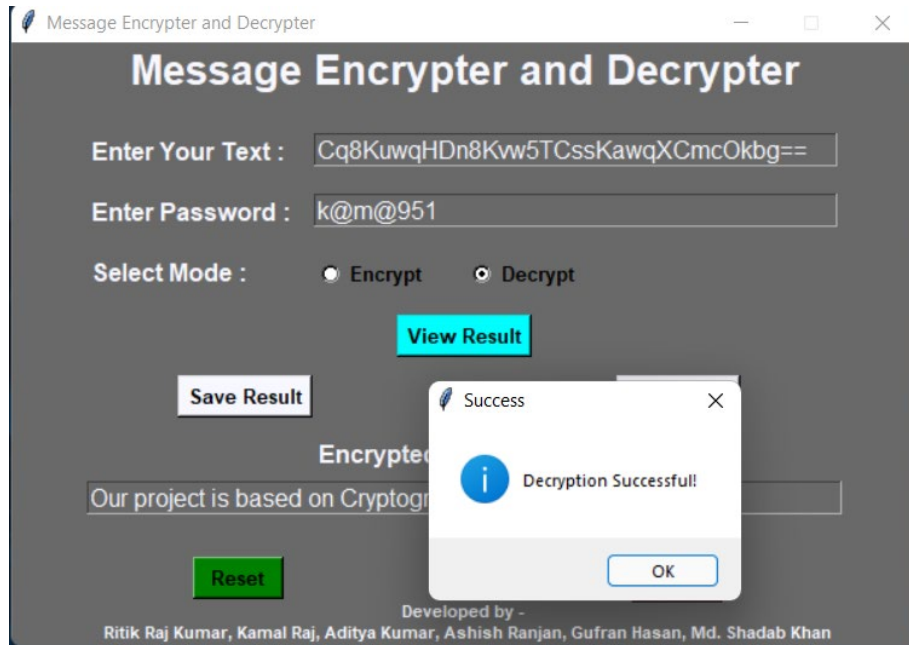
The screenshot shows the 'Message Encrypter and Decrypter' application window. The title bar reads 'Message Encrypter and Decrypter'. The main window has a dark gray background with the title 'Message Encrypter and Decrypter' in white. Below the title, there are three input fields: 'Enter Your Text :', 'Enter Password :', and 'Select Mode :'. The 'Select Mode' section has two radio buttons: 'Encrypt' (selected) and 'Decrypt'. Below these are three buttons: 'View Result' (cyan), 'Save Result' (white), and 'Select File' (white). At the bottom, there is a large text area labeled 'Encrypted/Decrypted Text :'. Below this are two buttons: 'Reset' (green) and 'Exit' (red). At the very bottom, it says 'Developed by - Ritik Raj Kumar, Kamal Raj, Aditya Kumar, Ashish Ranjan, Gufran Hasan, Md. Shadab Khan'.

After entering encrypted message and password, and selecting mode (Decrypt):

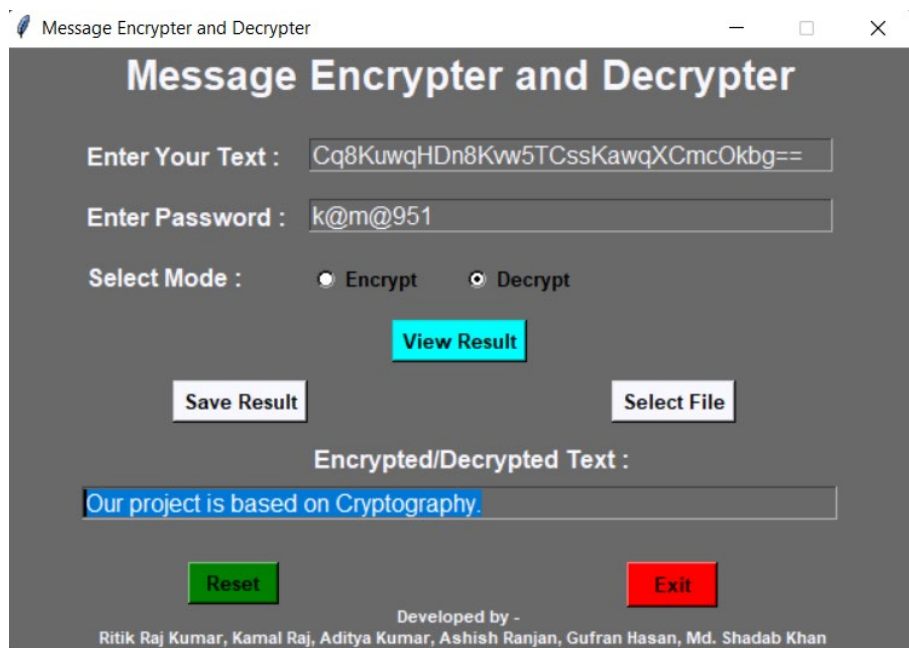


The screenshot shows the 'Message Encrypter and Decrypter' application window after data entry. The 'Enter Your Text' field now contains the encrypted message 'Cq8KuwqHDn8Kw5TCssKawqXCmcOkbg=='. The 'Enter Password' field contains 'k@m@951'. The 'Select Mode' section now has the 'Decrypt' radio button selected. The 'View Result' button remains cyan, while 'Save Result' and 'Select File' remain white. The 'Encrypted/Decrypted Text' area is still empty. The 'Reset' (green) and 'Exit' (red) buttons are at the bottom. The footer text remains the same: 'Developed by - Ritik Raj Kumar, Kamal Raj, Aditya Kumar, Ashish Ranjan, Gufran Hasan, Md. Shadab Khan'.

After clicking *View Result* button:



Decrypted Message:



Conclusion

We have successfully developed a program that encrypts and decrypts a message. This will help in minimizing the problem of data theft and leaks of other sensitive information. The message that we obtained after encryption is very safe and no one can know the original message. So, this file can be sent over a network without worrying.

GitHub Link: <https://bit.ly/33oTGg0>

References

1. <https://bit.ly/3nhuDIz>
2. <https://bit.ly/3neh8Uj>