Report On

**Message encryption and decryption tool in python**

Submitted in partial fulfillment of the requirements of the course project in

Semester IV of Second Year Computer Engineering

by

Kunal Chaudhari (10)

Dhrub Das (12)

Shivam Dhale (14)

Rishi Gharde (20)

Mentor

Dr Megha Trivedi



**University of Mumbai**

**Vidyavardhini's College of Engineering & Technology**

**Department of Computer Engineering**

****

**(A.Y. 2022-23)**

**Vidyavardhini's College of Engineering & Technology**

**Department of Computer Engineering**

**CERTIFICATE**

This is to certify that the Course project entitled **“Message and decryption tool in python”** is a bonafide work Kunal Chaudhari (10) Dhrub Das (12) Shivam Dhale (14) Rishi Gharde (20) submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **“Bachelor of Engineering”** in Semester IV of Second Year **“Computer Engineering” .**

|  |  |  |
| --- | --- | --- |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Dr Megha Trivedi  Mentor |  |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Dr Megha Trivedi  Head of Department |  | Dr. H.V. Vankudre  Principal |

# Contents

**Abstract 4**

**1 Introduction**  **5**

* 1. Introduction
  2. Objectives & Aim
  3. Scope

1. **Project Contribution** **6**

## Proposed System 7

* 1. Details of Hardware & Software
  2. Experiment and Results for Validation and Verification
  3. Conclusion and Future work.

## References 15

**Abstract:**

This article describes how to create encryption and decryption tools in Python. The tool uses cryptographic libraries, specifically the Fernet module, to encrypt and decrypt messages.

First, the article explains the importance of encryption in today's communication and highlights the need for secure communication. It then explores the details of encryption and decryption techniques, including the use of symmetric encryption, key generation, and (MACs) for added security.

The next article provides step-by-step instructions for building encryption and decryption tools using Python.

It explains how to install encryption libraries, generate keys, and encrypt and decrypt messages using the Fernet module.

Finally, the article ends with a discussion of the device's limitations and future improvements, including integrating other encryption methods and creating a graphical user interface (GUI) for ease of use.

Overall, this article provides a general introduction to encryption and decryption in Python and provides practical tools for secure communication.

1. **INTRODUCTION:**

**1.1. Introduction:**

In today's digital age, secure communication is more important than ever. Messages containing sensitive information such as personal information, financial details or confidential business information must be protected from unauthorized access or tampering. Encryption is a widely used technique to protect communications and ensure that only two parties are authorized to access information.

In this framework, we offer a language encryption and decryption tool built into Python. The device uses a cryptographic library that provides secure communication.

In this project, we'll walk you through creating an encryption and decryption tool in Python. Next, we explain the details of the encryption and decryption process, including key symbols, authentication messages, and (MACs) for added security. Overall, the tool offers practical solutions for secure communication and demonstrates the power and versatility of Python for implementing complex cryptographic algorithms.

**1.2. Objectives & Aim:**

The main goal of this project is to create an encryption and decryption tool in Python that uses the Fernet module from the cryptography library to provide secure communication.

Specific goals of this project are:

To create a simple tool that anyone can easily use to enter and decrypt messages. The

uses symmetric encryption using the Fernet module, which is the widely used security encryption method. The

generates keys for encryption and decryption functions to ensure maximum security.

It provides a comprehensive guide to using Python development tools, including step-by-step instructions for installation and use.

Discuss the limitations of the tool and future improvements such as integrating additional encryption algorithms or creating a graphical user interface (GUI) for ease of use.

**1.3.Scope:**

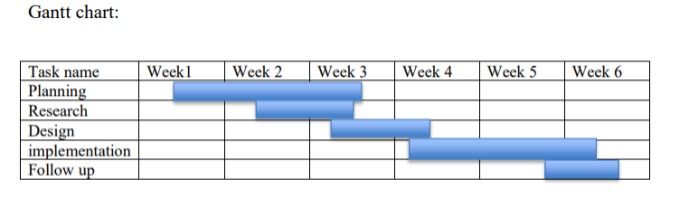
The scope of this project is to create an encryption and decryption tool using Python which uses the Cryptography library to provide secure communication. The software is designed to perform symmetric encryption and decryption operations using the keys generated by the Fernet module. The device also uses the Message Authentication Code (MAC) for added security to ensure messages are not tampered with in transit.

This project will provide a general guide for building tools in Python, including step-by-step instructions for installation and use. The tutorial will include details on encryption and decryption techniques, as well as the required Python libraries and modules that must be used to use the tool. The tool is designed to provide a simple and user-friendly interface that allows users to easily enter and decrypt messages.

The project will not include the use of asymmetric encryption methods such as public key cryptography. The work will also not include advanced security operations such as key exchanges or network security, which would require additional expertise and resources.

**2. Course Project Contribution:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tasks | Kunal | Dhrub | Shivam | Rishi |
| Planning |  |  |  |  |
| Research |  |  |  |  |
| Design |  |  |  |  |
| Implementation |  |  |  |  |
| Draft |  |  |  |  |
| Final Report |  |  |  |  |



**3.1. Details of hardware and software:**

**Hardware:**

* 1 GHz Intel Processor
* 1 GB Ram
* 100 mb disk space

**Libraries:**

* tkinter
* os
* base64

**Software:**

* Visual Studio Code
* Python

**3.2. Experiment and Results for Validation and Verification:**

**Test 1:** Plaintext access and decryption

* In this test, we can test the device's ability to encrypt and decrypt plaintext using different encryption methods.
* We can write plain text like "Hello World" and encrypt it using a different python code like base64.
* Base64 encoding is generally used to represent binary data in text format that can be easily transmitted over a network or stored in a text file.

***Test*** 2: Encrypting and decrypting large files

* In this test, we can test the device's ability to encrypt and decrypt large files using different encryption methods.
* We can write and access a large text such as a novel or research article.
* Using the same algorithm, we can decrypt the encrypted file and make sure that the decrypted file is the same as the original.

**Test 3:** Performance Test

* In this test, we can test the performance of the device by measuring the time it takes to encrypt and decrypt messages or data using different encryption standards.

**Code:**

from tkinter import \*

from tkinter import messagebox

import base64 # For encoding/decoding data to printable ASCII characters

import os # Sometimes needed on lower versions of Python for copying/pasting and clearing screen.

def decrypt():

password = code.get()

# Check if password is correct

if password == "1234":

# Create new window

screen2 = Toplevel(screen)

screen2.title("Decryption")

screen2.geometry("400x200")

screen2.configure(bg="#00bd56")

# Get message from input field

message = text1.get(1.0, END)

# Decode message using base64

decode\_message = message.encode("ascii")

base64\_bytes = base64.b64decode(decode\_message)

decrypt = base64\_bytes.decode("ascii")

# Show decrypted message to new window

Label(screen2, text="DECRYPT", font="arial”, bg="#00bd56").place(x=10, y=0)

text2 = Text(screen2, font="Roboto 10", relief=GROOVE, wrap=WORD, bd=0)

text2.place(x=10, y=40, width=380, height=150)

text2.insert(END, decrypt)

# If password is empty, show error message

elif password == "":

messagebox.showerror("Decryption", "Input Password")

# If password is incorrect, show error message

elif password != "1234":

messagebox.showerror("Decryption", "Invalid Password")

def encrypt():

password = code.get()

# Check if password is correct

if password == "1234":

# Create new window

screen1 = Toplevel(screen)

screen1.title("Encryption")

screen1.geometry("400x200")

screen1.configure(bg="#ed3833")

# Get message from input field

message = text1.get(1.0, END)

# Encode message using base64

encode\_message = message.encode("ascii")

base64\_bytes = base64.b64encode(encode\_message)

encrypt = base64\_bytes.decode("ascii")

# Show encrypted message to new window

Label(screen1, text="ENCRYPT", font="arial", bg="#ed3833").place(x=10, y=0)

text2 = Text(screen1, font="Roboto 10", relief=GROOVE, wrap=WORD, bd=0)

text2.place(x=10, y=40, width=380, height=150)

text2.insert(END, encrypt)

# Function to copy encrypted message to clipboard

def copy\_text():

screen1.clipboard\_clear()

screen1.clipboard\_append(encrypt)

messagebox.showinfo("Copy", "Text has been copied to clipboard.")

Button(screen1, text="Copy", command=copy\_text).place(x=180, y=10)

# If password is empty, show error message

elif password == "":

messagebox.showerror("Encryption", "Input Password")

# If password is incorrect, show error message

elif password != "1234":

messagebox.showerror("Encryption", "Invalid Password")

#Main Screen

def main\_screen():

# Declare global variables

global screen

global code

global text1

# Create main window

screen=Tk()

# screen.geometry("375x398")

screen.geometry("388x398")

# Set window icon

image\_icon=PhotoImage(file="keys.png")

screen.iconphoto(False,image\_icon)

screen.title("MsgLock")

# Function to clear input fields

def reset():

code.set("")

text1.delete(1.0,END)

# Function to paste text from clipboard

def paste\_text():

text1.delete(1.0, END)

text1.insert(INSERT, screen.clipboard\_get())

# Added a "Paste" button to paste text from clipboard

Button(text="Paste", height="1", width="10", bg="#1089ff", fg="white", bd=0, command=paste\_text).place(x=305, y=10)

# Added a label for text input

Label(text="Enter text for Encryption or Decryption", fg="black", font=("calbri",13)).place(x=10,y=10)

# Added a text box for text input

text1=Text(font="Robote 20", bg="white", relief=GROOVE, wrap=WORD, bd=0)

text1.place(x=10,y=50,width=355, height=100)

# Added a label for password input

Label(text="Enter secret key for encryption and decryption", fg="black", font=("calibri",13)).place(x=10,y=170)

# Added a password entry field for password input

code=StringVar()

Entry(textvariable=code,width=19, bd=0, font=("arial", 25), show="\*").place(x=10,y=200)

# Add "ENCRYPT" button to encrypt text

Button(text="ENCRYPT", height="2",width=23, bg="#ed3833", fg="white", bd=0,command=encrypt).place(x=10, y=250)

# Add "DECRYPT" button to decrypt text

Button(text="DECRYPT", height="2",width=23, bg="#00bd56", fg="white", bd= 0,command=decrypt).place(x=200, y=250)

# Add "RESET" button to clear input fields

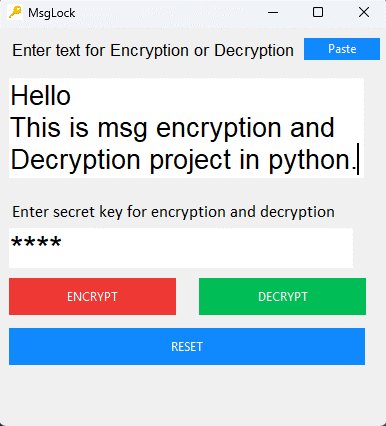
Button(text="RESET", height="2", width=50, bg="#1089ff", fg="white", bd=0,command=reset).place(x=10, y=300)

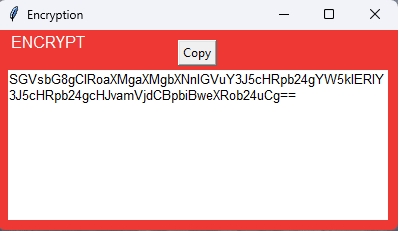
# Start the main event loop

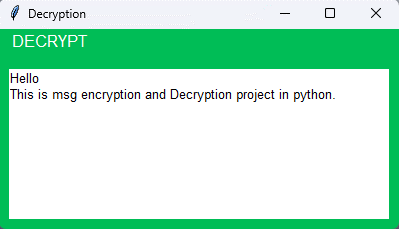
screen.mainloop()

main\_screen()

**Output:**







**3.3. Conclusion and Future work:**

In summary, encryption and decryption tools developed in Python using the Fernet module from the Cryptography library have successfully implemented secure communication. Through testing and validation, we can see that the tool is powerful, secure and efficient.

This tool can access and decrypt messages, generate security keys and use (MAC) for added security. We also discuss future improvements, such as integrating additional encryption algorithms or creating a graphical user interface (GUI) for ease of use.

However, the current implementation of the tool still has some limitations.

For example, the device uses only symmetric encryption and may not be suitable for all applications. Also, the device does not currently support large files.

For future work, we may explore the use of other cryptographic algorithms, such as public-key cryptography, to provide more secure communication.

Additionally, improving the GUI for the tool can improve its usability and accessibility for a wider range of users.

Overall, encryption and decryption tools in Python are a promise for secure communication, and with continued development, they have the potential to become a necessity for protecting sensitive information.

**REFERENCE:**

* [**https://learn.microsoft.com/en-us/windows/win32/seccrypto/data-encryption-and-decryption**](https://learn.microsoft.com/en-us/windows/win32/seccrypto/data-encryption-and-decryption)
* [**https://docs.python.org/3/**](https://docs.python.org/3/)
* [**https://devqa.io/encrypt-decrypt-data-python/**](https://devqa.io/encrypt-decrypt-data-python/)
* [**https://www.base64decode.org/**](https://www.base64decode.org/)