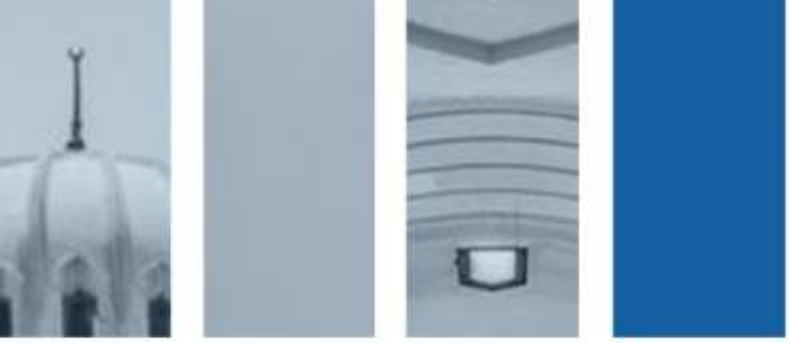


Secure-Email with Key Distribution Project

Program: CESS

Course Code: CSE451

Course Name: Computer and Network Security

Examination Committee

Dr. Ayman Bahaa

TA Eng. Waleed

TA Eng. Ahmed

Student Info:

Mohamed Sayed Awwad 18P7298

Mahmoud morad   18P6555

Ain Shams University Faculty of Engineering

Spring Semester – 2023

Contents

Idea

a) Project idea

The idea of the project is to achieve a secure email exchange with an end to end encryption using a central key distribution center

b) Solution idea

Every message transmitted from a key distribution server (KDS) will use a different dummy key sent with a mail, but each mail will use a unique central key. AES-128 will be used to encrypt the message itself.

Requirements

Use 4 python files to:

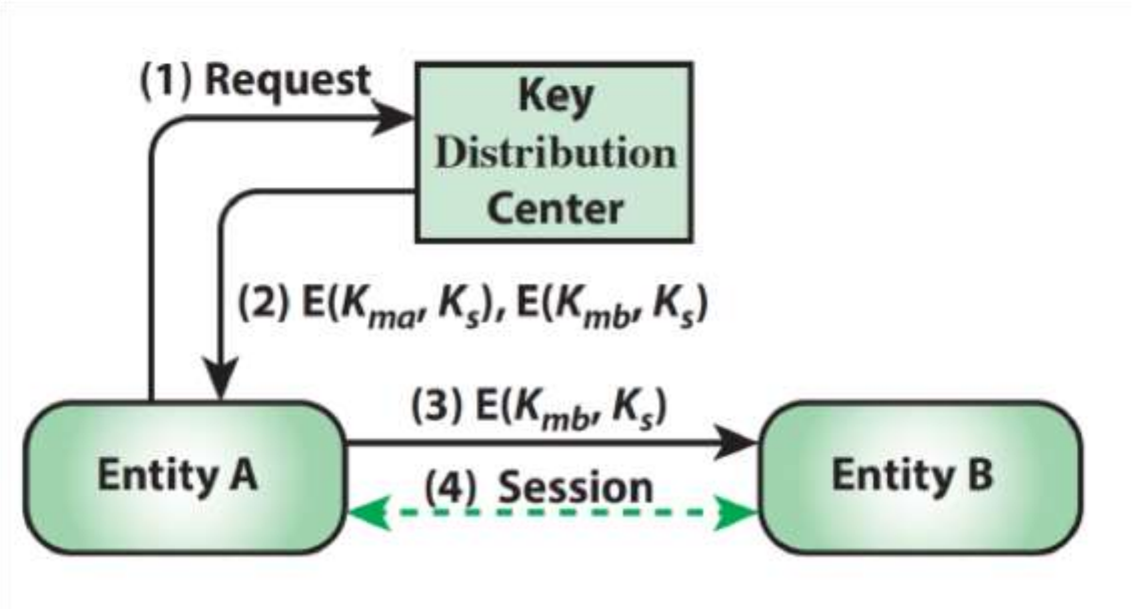
1) Send the mail
2) Encrypt the messages
3) Prepare the server
4) Decrypt the message.

Design

a) How it works

1. A separate server is implemented to work as follows(by **cryptodome** library):
   a. Each user is registered on the server with his email.
   b. Each user has a master 128-bit secret key only known to the user and the KDS.
   c. The KDS when receives a request, it generates a key and sends 2 different copies of the key to the requesting end, each copy is encrypted by the master key of the requester and the recipient respectively.
2. When the sending client is done with key, it just puts it as the second attachment

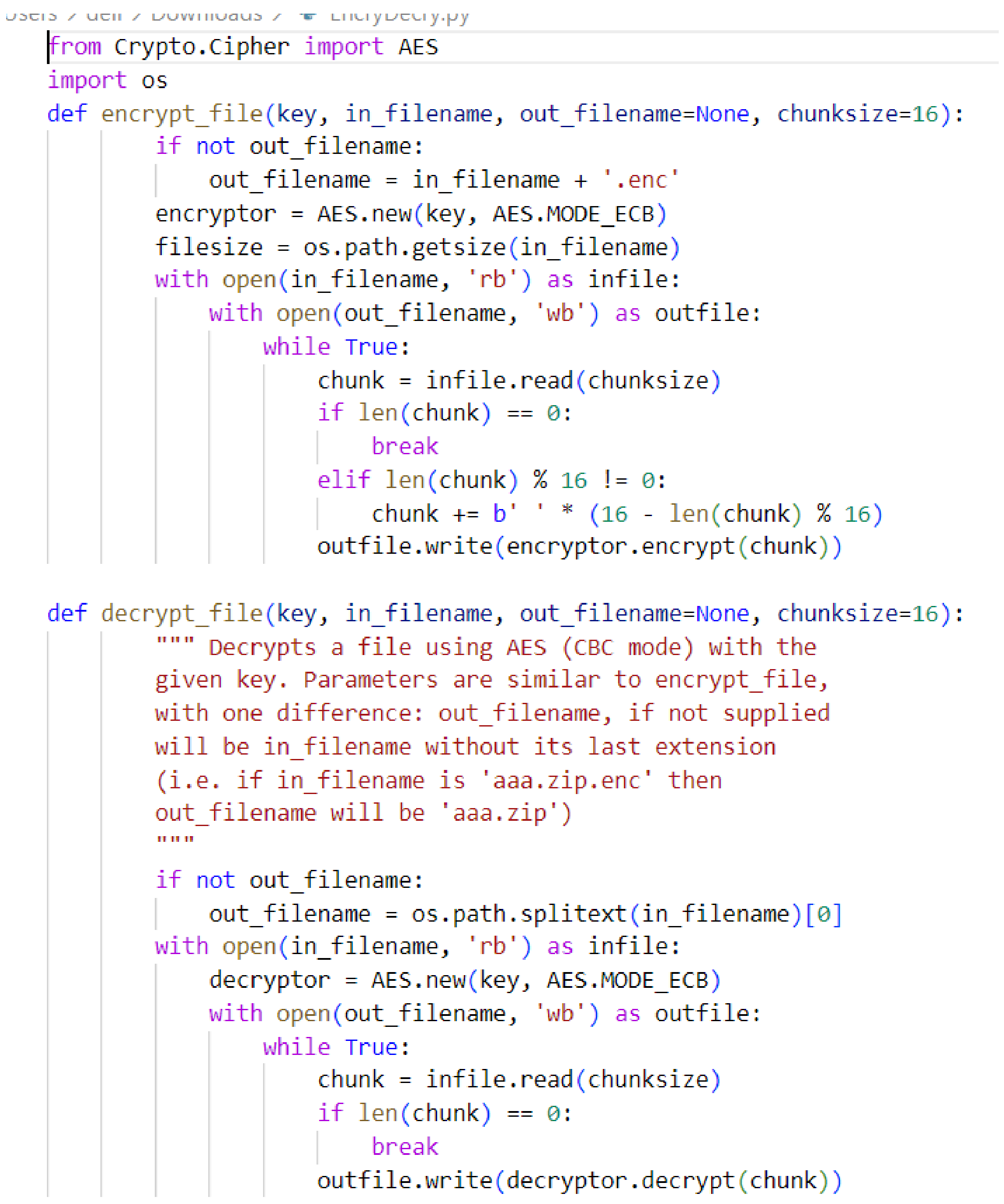
b) Architecture



Implementation

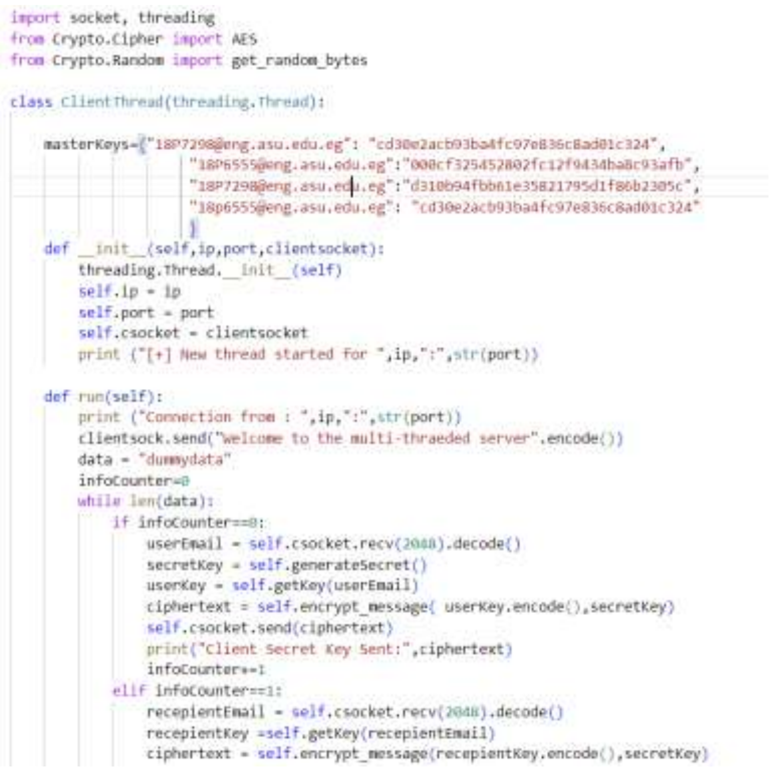
a) EncDec.py

Encryption and decryption of the message using AES

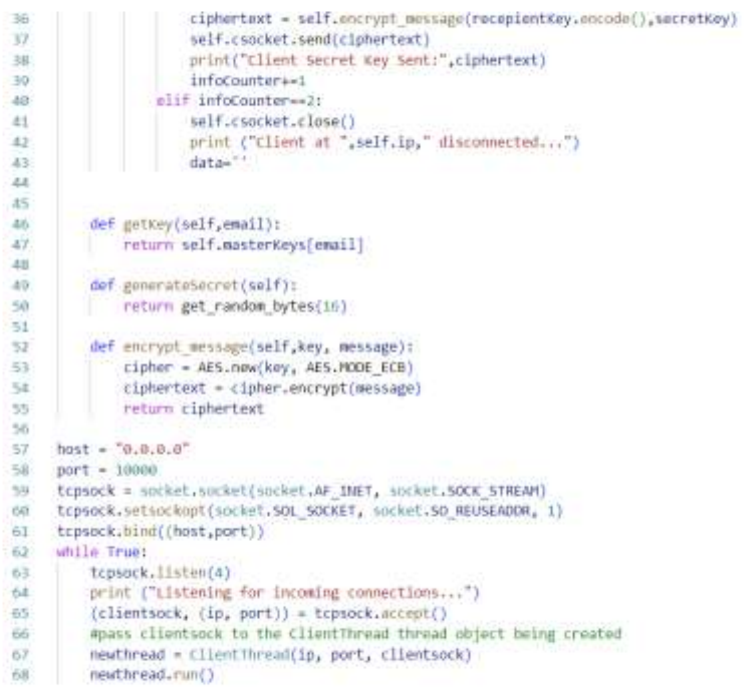


```python
from Crypto.Cipher import AES
import os
def encrypt_file(key, in_filename, out_filename=None, chunksize=16):
        if not out_filename:
            out_filename = in_filename + '.enc'
        encryptor = AES.new(key, AES.MODE_ECB)
        filesize = os.path.getsize(in_filename)
        with open(in_filename, 'rb') as infile:
            with open(out_filename, 'wb') as outfile:
                while True:
                    chunk = infile.read(chunksize)
                    if len(chunk) == 0:
                        break
                    elif len(chunk) % 16 != 0:
                        chunk += b' ' * (16 - len(chunk) % 16)
                    outfile.write(encryptor.encrypt(chunk))

def decrypt_file(key, in_filename, out_filename=None, chunksize=16):
        """ Decrypts a file using AES (CBC mode) with the
        given key. Parameters are similar to encrypt_file,
        with one difference: out_filename, if not supplied
        will be in_filename without its last extension
        (i.e. if in_filename is 'aaa.zip.enc' then
        out_filename will be 'aaa.zip')
        """
        if not out_filename:
            out_filename = os.path.splitext(in_filename)[0]
        with open(in_filename, 'rb') as infile:
            decryptor = AES.new(key, AES.MODE_ECB)
            with open(out_filename, 'wb') as outfile:
                while True:
                    chunk = infile.read(chunksize)
                    if len(chunk) == 0:
                        break
                    outfile.write(decryptor.decrypt(chunk))
```

b) server.py

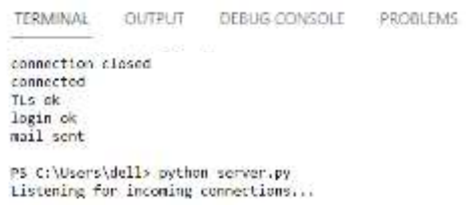


```python
import socket, threading
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes

class ClientThread(threading.Thread):

    masterKeys={"18P7298@eng.asu.edu.eg": "cd30e2acb93ba4fc97e836c8ad01c324",
                "18P6555@eng.asu.edu.eg":"000cf325452802fc12f9434ba8c93afb",
                "18P7298@eng.asu.edu.eg":"d310b94fbb61e35821795d1f86b2305c",
                "18p6555@eng.asu.edu.eg": "cd30e2acb93ba4fc97e836c8ad01c324"
                }
    def __init__(self,ip,port,clientsocket):
        threading.Thread.__init__(self)
        self.ip = ip
        self.port = port
        self.csocket = clientsocket
        print ("[+] New thread started for ",ip,":",str(port))

    def run(self):
        print ("Connection from : ",ip,":",str(port))
        clientsock.send("welcome to the multi-thraeded server".encode())
        data = "dummydata"
        infoCounter=0
        while len(data):
            if infoCounter==0:
                userEmail = self.csocket.recv(2048).decode()
                secretKey = self.generateSecret()
                userKey = self.getKey(userEmail)
                ciphertext = self.encrypt_message( userKey.encode(),secretKey)
                self.csocket.send(ciphertext)
                print("Client Secret Key Sent:",ciphertext)
                infoCounter+=1
            elif infoCounter==1:
                recepientEmail = self.csocket.recv(2048).decode()
                recepientKey =self.getKey(recepientEmail)
                ciphertext = self.encrypt_message(recepientKey.encode(),secretKey)
```



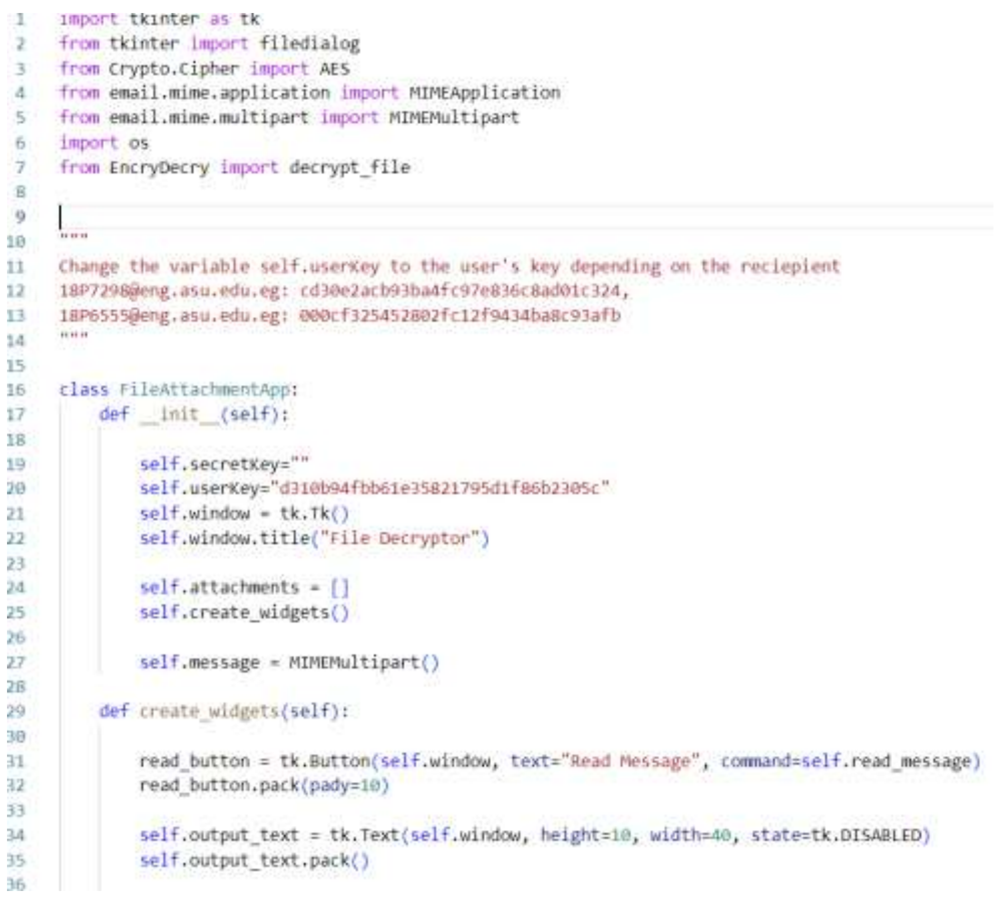
```python
                ciphertext = self.encrypt_message(recepientKey.encode(),secretKey)
                self.csocket.send(ciphertext)
                print("Client Secret Key Sent:",ciphertext)
                infoCounter+=1
            elif infoCounter==2:
                self.csocket.close()
                print ("Client at ",self.ip," disconnected...")
                data=''


    def getKey(self,email):
        return self.masterKeys[email]

    def generateSecret(self):
        return get_random_bytes(16)

    def encrypt_message(self,key, message):
        cipher = AES.new(key, AES.MODE_ECB)
        ciphertext = cipher.encrypt(message)
        return ciphertext

host = "0.0.0.0"
port = 10000
tcpsock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
tcpsock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
tcpsock.bind((host,port))
while True:
    tcpsock.listen(4)
    print ("Listening for incoming connections...")
    (clientsock, (ip, port)) = tcpsock.accept()
    #pass clientsock to the ClientThread thread object being created
    newthread = ClientThread(ip, port, clientsock)
    newthread.run()
```

Server is running

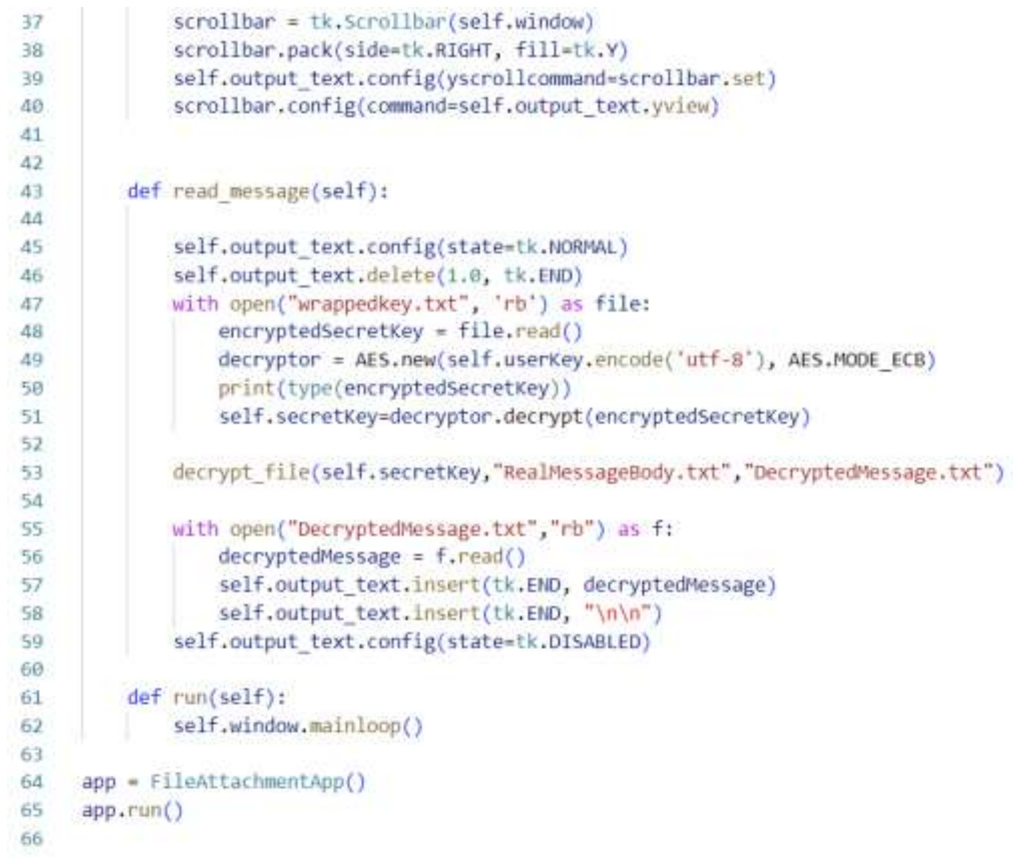


```
connection closed
connected
TLs ok
login ok
mail sent

PS C:\Users\dell> python server.py
Listening for incoming connections...
```

c) RecievedDecryptionTest.py



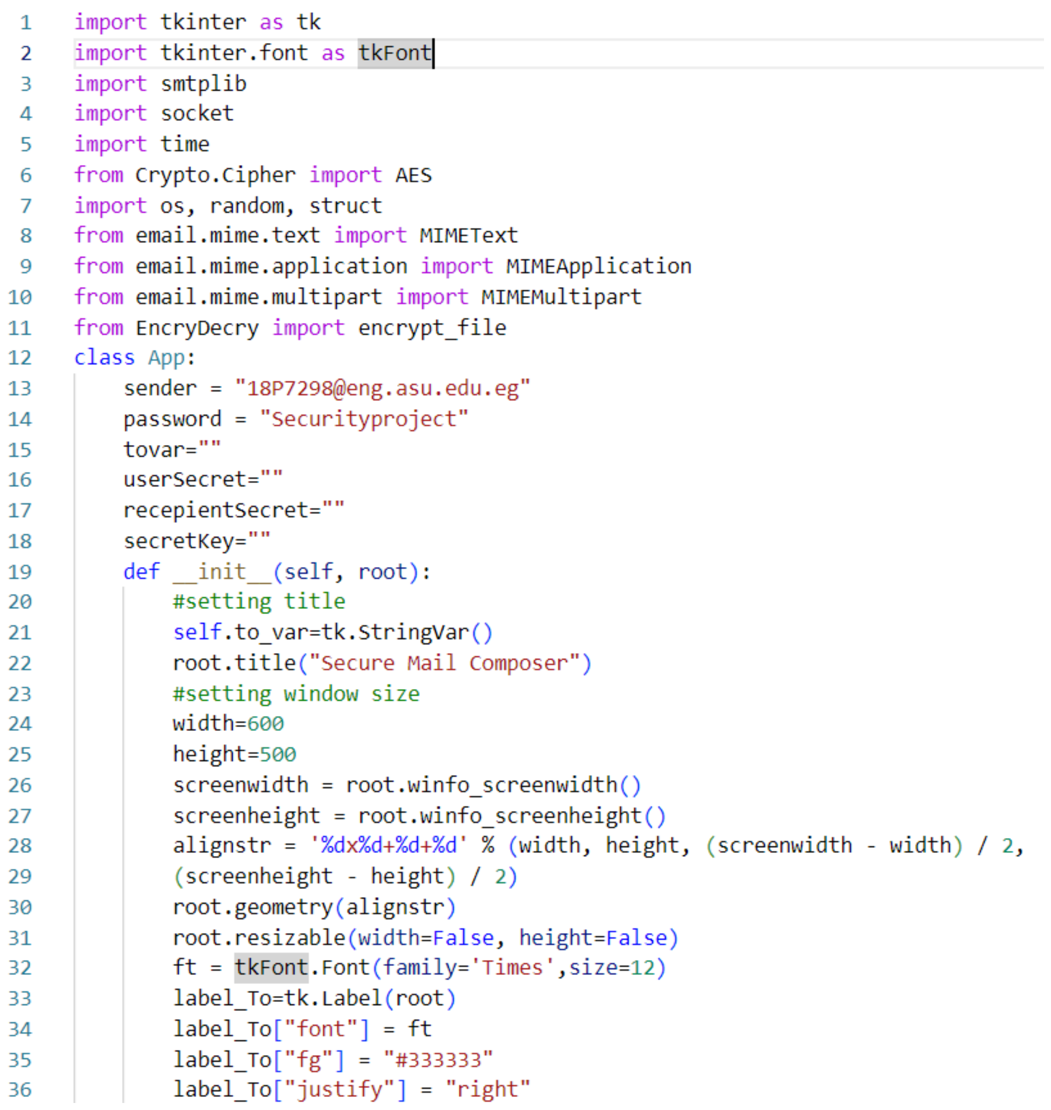
```python
import tkinter as tk
from tkinter import filedialog
from Crypto.Cipher import AES
from email.mime.application import MIMEApplication
from email.mime.multipart import MIMEMultipart
import os
from EncryDecry import decrypt_file


"""
Change the variable self.userKey to the user's key depending on the reciepient
18P7298@eng.asu.edu.eg: cd30e2acb93ba4fc97e836c8ad01c324,
18P6555@eng.asu.edu.eg: 000cf325452802fc12f9434ba8c93afb
"""

class FileAttachmentApp:
    def __init__(self):

        self.secretKey=""
        self.userKey="d310b94fbb61e35821795d1f86b2305c"
        self.window = tk.Tk()
        self.window.title("File Decryptor")

        self.attachments = []
        self.create_widgets()

        self.message = MIMEMultipart()

    def create_widgets(self):

        read_button = tk.Button(self.window, text="Read Message", command=self.read_message)
        read_button.pack(pady=10)

        self.output_text = tk.Text(self.window, height=10, width=40, state=tk.DISABLED)
        self.output_text.pack()

```



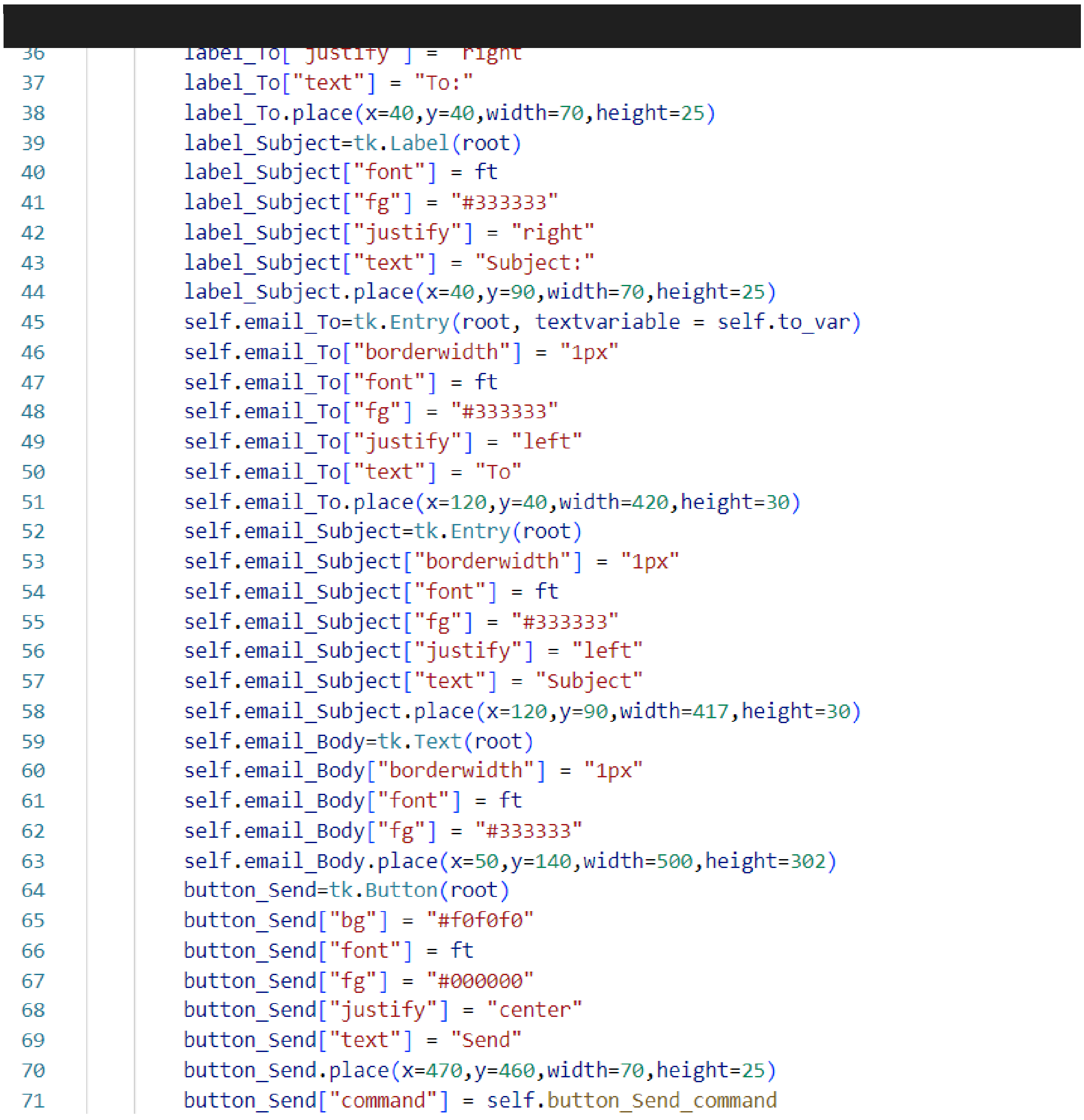
```python
        scrollbar = tk.Scrollbar(self.window)
        scrollbar.pack(side=tk.RIGHT, fill=tk.Y)
        self.output_text.config(yscrollcommand=scrollbar.set)
        scrollbar.config(command=self.output_text.yview)


    def read_message(self):

        self.output_text.config(state=tk.NORMAL)
        self.output_text.delete(1.0, tk.END)
        with open("wrappedkey.txt", 'rb') as file:
            encryptedSecretKey = file.read()
            decryptor = AES.new(self.userKey.encode('utf-8'), AES.MODE_ECB)
            print(type(encryptedSecretKey))
            self.secretKey=decryptor.decrypt(encryptedSecretKey)

        decrypt_file(self.secretKey,"RealMessageBody.txt","DecryptedMessage.txt")

        with open("DecryptedMessage.txt","rb") as f:
            decryptedMessage = f.read()
            self.output_text.insert(tk.END, decryptedMessage)
            self.output_text.insert(tk.END, "\n\n")
        self.output_text.config(state=tk.DISABLED)

    def run(self):
        self.window.mainloop()

app = FileAttachmentApp()
app.run()

```

d) ExApplication.py

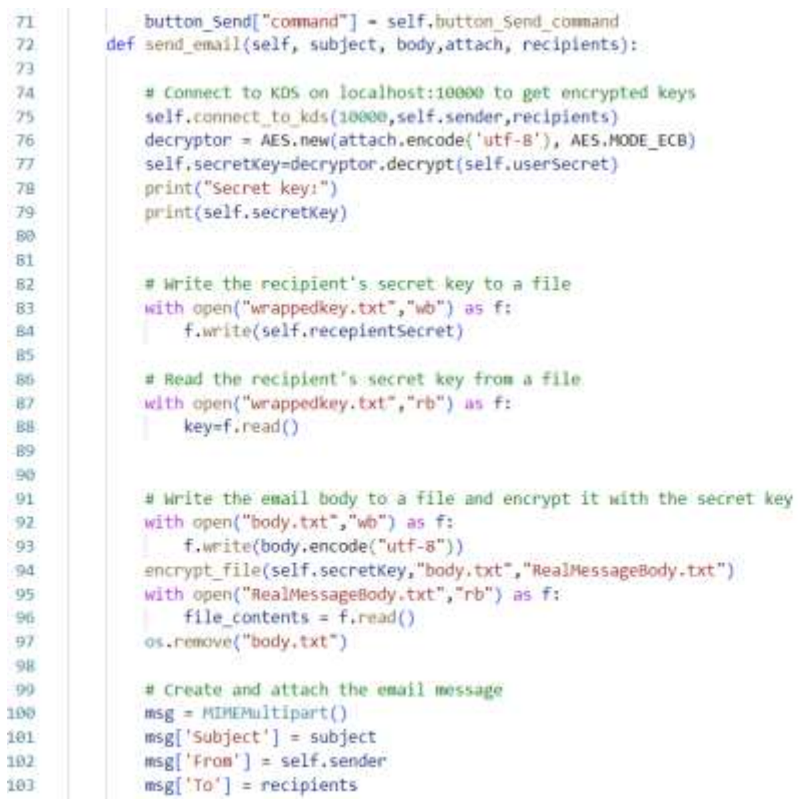
This file is used to send the email message and key that is received from the KDS server



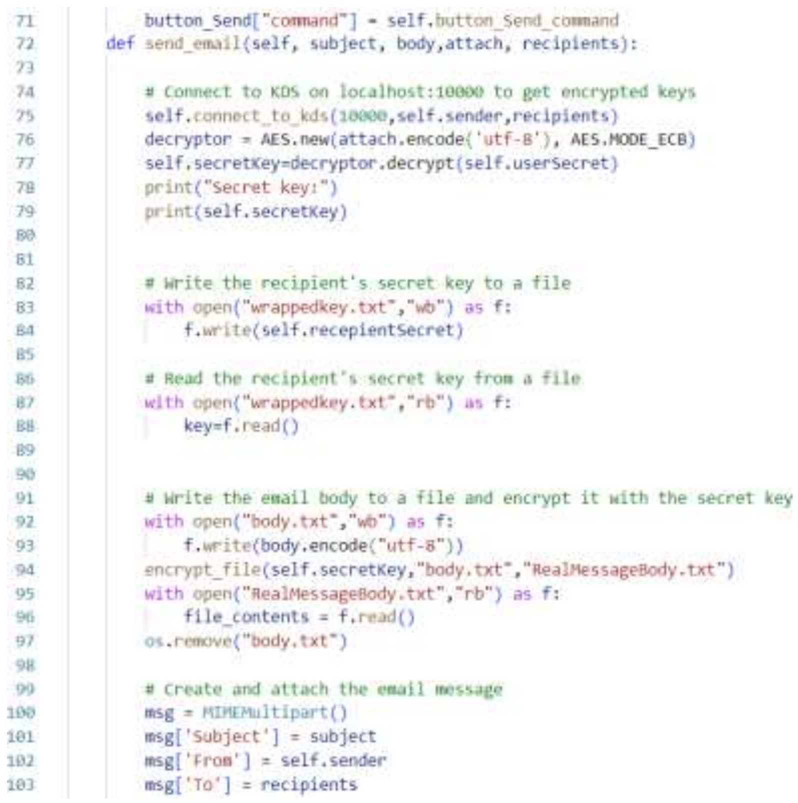
```python
import tkinter as tk
import tkinter.font as tkFont
import smtplib
import socket
import time
from Crypto.Cipher import AES
import os, random, struct
from email.mime.text import MIMEText
from email.mime.application import MIMEApplication
from email.mime.multipart import MIMEMultipart
from EncryDecry import encrypt_file
class App:
    sender = "18P7298@eng.asu.edu.eg"
    password = "Securityproject"
    tovar=""
    userSecret=""
    recepientSecret=""
    secretKey=""
    def __init__(self, root):
        #setting title
        self.to_var=tk.StringVar()
        root.title("Secure Mail Composer")
        #setting window size
        width=600
        height=500
        screenwidth = root.winfo_screenwidth()
        screenheight = root.winfo_screenheight()
        alignstr = '%dx%d+%d+%d' % (width, height, (screenwidth - width) / 2,
        (screenheight - height) / 2)
        root.geometry(alignstr)
        root.resizable(width=False, height=False)
        ft = tkFont.Font(family='Times',size=12)
        label_To=tk.Label(root)
        label_To["font"] = ft
        label_To["fg"] = "#333333"
        label_To["justify"] = "right"
```



```python
        label_To["justify"] = "right"
        label_To["text"] = "To:"
        label_To.place(x=40,y=40,width=70,height=25)
        label_Subject=tk.Label(root)
        label_Subject["font"] = ft
        label_Subject["fg"] = "#333333"
        label_Subject["justify"] = "right"
        label_Subject["text"] = "Subject:"
        label_Subject.place(x=40,y=90,width=70,height=25)
        self.email_To=tk.Entry(root, textvariable = self.to_var)
        self.email_To["borderwidth"] = "1px"
        self.email_To["font"] = ft
        self.email_To["fg"] = "#333333"
        self.email_To["justify"] = "left"
        self.email_To["text"] = "To"
        self.email_To.place(x=120,y=40,width=420,height=30)
        self.email_Subject=tk.Entry(root)
        self.email_Subject["borderwidth"] = "1px"
        self.email_Subject["font"] = ft
        self.email_Subject["fg"] = "#333333"
        self.email_Subject["justify"] = "left"
        self.email_Subject["text"] = "Subject"
        self.email_Subject.place(x=120,y=90,width=417,height=30)
        self.email_Body=tk.Text(root)
        self.email_Body["borderwidth"] = "1px"
        self.email_Body["font"] = ft
        self.email_Body["fg"] = "#333333"
        self.email_Body.place(x=50,y=140,width=500,height=302)
        button_Send=tk.Button(root)
        button_Send["bg"] = "#f0f0f0"
        button_Send["font"] = ft
        button_Send["fg"] = "#000000"
        button_Send["justify"] = "center"
        button_Send["text"] = "Send"
        button_Send.place(x=470,y=460,width=70,height=25)
        button_Send["command"] = self.button_Send_command
```

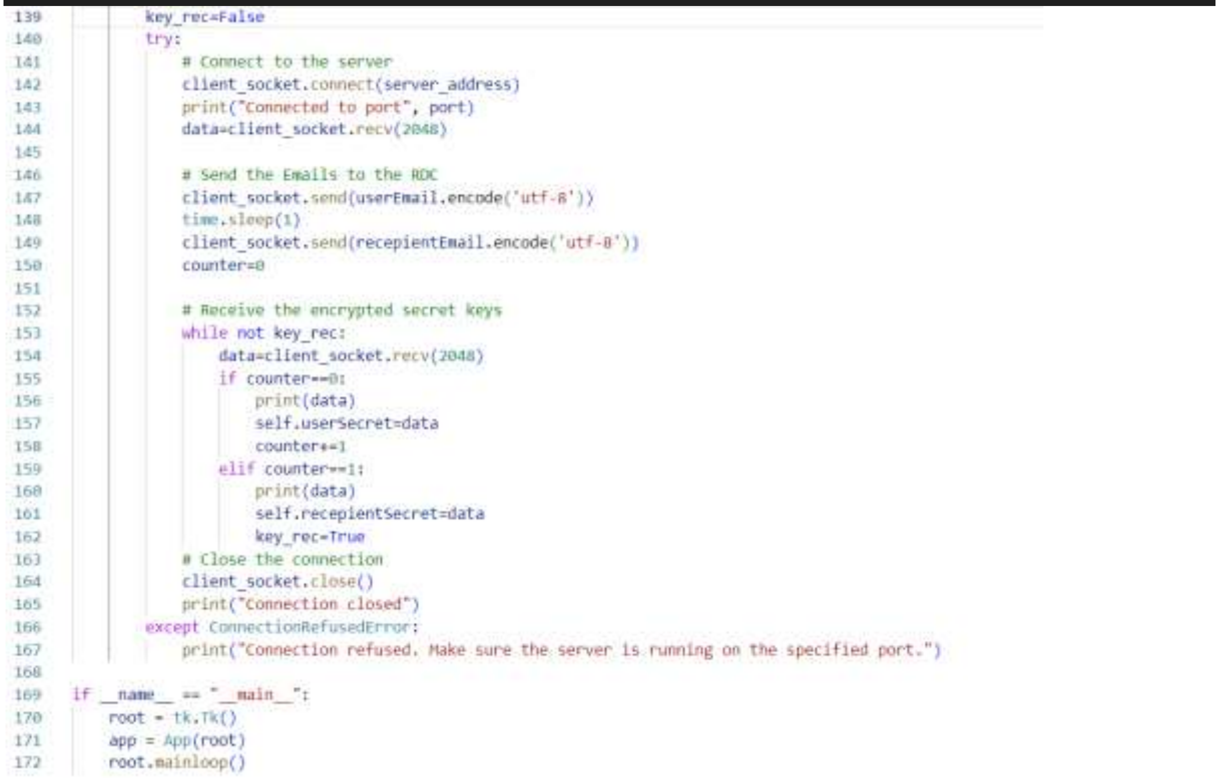


```python
        button_Send["command"] = self.button_Send_command
    def send_email(self, subject, body,attach, recipients):

        # Connect to KDS on localhost:10000 to get encrypted keys
        self.connect_to_kds(10000,self.sender,recipients)
        decryptor = AES.new(attach.encode('utf-8'), AES.MODE_ECB)
        self.secretKey=decryptor.decrypt(self.userSecret)
        print("Secret key:")
        print(self.secretKey)


        # Write the recipient's secret key to a file
        with open("wrappedkey.txt","wb") as f:
            f.write(self.recepientSecret)

        # Read the recipient's secret key from a file
        with open("wrappedkey.txt","rb") as f:
            key=f.read()


        # Write the email body to a file and encrypt it with the secret key
        with open("body.txt","wb") as f:
            f.write(body.encode("utf-8"))
        encrypt_file(self.secretKey,"body.txt","RealMessageBody.txt")
        with open("RealMessageBody.txt","rb") as f:
            file_contents = f.read()
        os.remove("body.txt")

        # Create and attach the email message
        msg = MIMEMultipart()
        msg['Subject'] = subject
        msg['From'] = self.sender
        msg['To'] = recipients
```



```python
        button_Send["command"] = self.button_Send_command
    def send_email(self, subject, body,attach, recipients):

        # Connect to KDS on localhost:10000 to get encrypted keys
        self.connect_to_kds(10000,self.sender,recipients)
        decryptor = AES.new(attach.encode('utf-8'), AES.MODE_ECB)
        self.secretKey=decryptor.decrypt(self.userSecret)
        print("Secret key:")
        print(self.secretKey)


        # Write the recipient's secret key to a file
        with open("wrappedkey.txt","wb") as f:
            f.write(self.recepientSecret)

        # Read the recipient's secret key from a file
        with open("wrappedkey.txt","rb") as f:
            key=f.read()


        # Write the email body to a file and encrypt it with the secret key
        with open("body.txt","wb") as f:
            f.write(body.encode("utf-8"))
        encrypt_file(self.secretKey,"body.txt","RealMessageBody.txt")
        with open("RealMessageBody.txt","rb") as f:
            file_contents = f.read()
        os.remove("body.txt")

        # Create and attach the email message
        msg = MIMEMultipart()
        msg['Subject'] = subject
        msg['From'] = self.sender
        msg['To'] = recipients
```

[redacted]



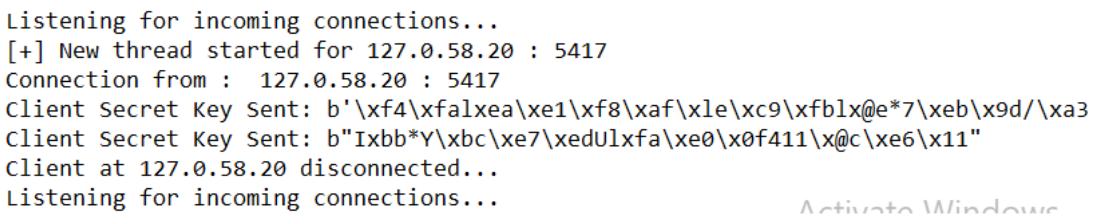
```
        key_rec=False
        try:
            # Connect to the server
            client_socket.connect(server_address)
            print("Connected to port", port)
            data=client_socket.recv(2048)

            # Send the Emails to the RDC
            client_socket.send(userEmail.encode('utf-8'))
            time.sleep(1)
            client_socket.send(recepientEmail.encode('utf-8'))
            counter=0

            # Receive the encrypted secret keys
            while not key_rec:
                data=client_socket.recv(2048)
                if counter==0:
                    print(data)
                    self.userSecret=data
                    counter+=1
                elif counter==1:
                    print(data)
                    self.recepientSecret=data
                    key_rec=True
            # Close the connection
            client_socket.close()
            print("Connection closed")
        except ConnectionRefusedError:
            print("Connection refused. Make sure the server is running on the specified port.")

if __name__ == "__main__":
    root = tk.Tk()
    app = App(root)
    root.mainloop()
```

Test Cases

Test case 1

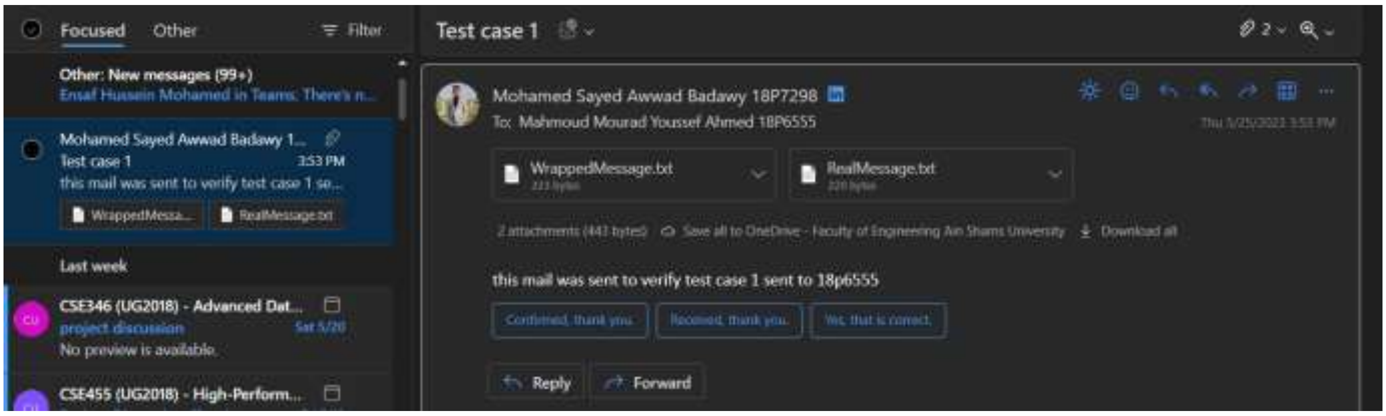
Send a message to 18p6555 and decrypt with the correct key associated with this ID.

Mail sent to 18p6555



```
Listening for incoming connections...
[+] New thread started for 127.0.58.20 : 5417
Connection from :  127.0.58.20 : 5417
Client Secret Key Sent: b'\xf4\xfalxea\xe1\xf8\xaf\xle\xc9\xfblx@e*7\xeb\x9d/\xa3
Client Secret Key Sent: b"Ixbb*Y\xbc\xe7\xedUlxfa\xe0\x0f411\x@c\xe6\x11"
Client at 127.0.58.20 disconnected...
Listening for incoming connections...
```

Mail received



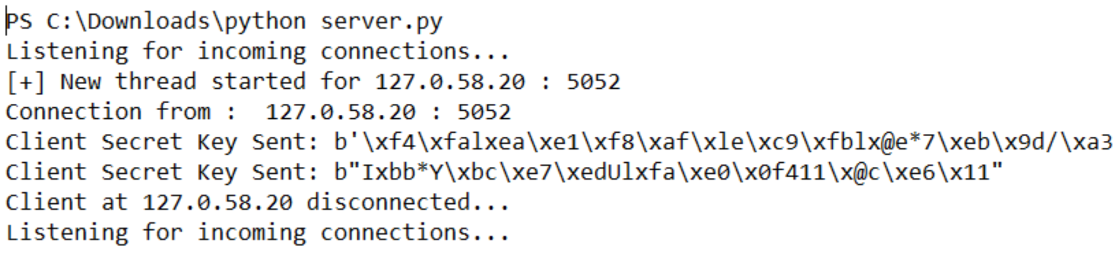
Focused
Other
Filter
Other: New messages (99+)
Ensaf Hussein Mohamed in Teams: There's n...
Mohamed Sayed Awwad Badawy 1...
Test case 1
3:53 PM
this mail was sent to verify test case 1 se...
WrappedMessa...
RealMessage.txt
Last week
CSE346 (UG2018) - Advanced Dat...
project discussion
Sat 5/20
No preview is available.
CSE455 (UG2018) - High-Perform...
Test case 1
Mohamed Sayed Awwad Badawy 18P7298
To: Mahmoud Mourad Youssef Ahmed 18P6555
WrappedMessage.txt
RealMessage.txt
2 attachments (443 bytes)
Save all to OneDrive - Faculty of Engineering Ain Shams University
Download all
Confirmed, thank you.
Received, thank you.
Yes, that is correct.
Reply
Forward

this mail was sent to verify test case 1 sent to 18p6555

Test case 2

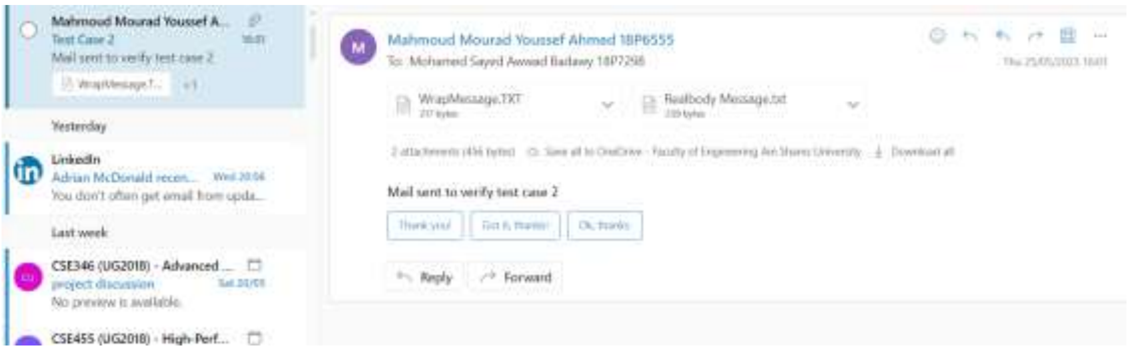
Send a message to 18P7298 and decrypt with the correct key associated with this ID.

Mail sent



```
PS C:\Downloads\python server.py
Listening for incoming connections...
[+] New thread started for 127.0.58.20 : 5052
Connection from :  127.0.58.20 : 5052
Client Secret Key Sent: b'\xf4\xfalxea\xe1\xf8\xaf\xle\xc9\xfblx@e*7\xeb\x9d/\xa3
Client Secret Key Sent: b"Ixbb*Y\xbc\xe7\xedUlxfa\xe0\x0f411\x@c\xe6\x11"
Client at 127.0.58.20 disconnected...
Listening for incoming connections...
```

Mail received



Mahmoud Mourad Youssef A...
Test Case 2 16:01
Mail sent to verify test case 2
WrapMessage.T... +1

Yesterday

LinkedIn
Adrian McDonald recen... Wed 20:06
You don't often get email from upda...

Last week

CSE346 (UG2018) - Advanced ...
project discussion Sat 20/05
No preview is available.

CSE455 (UG2018) - High-Perf...

Mahmoud Mourad Youssef Ahmed 18P6555
To: Mohamed Sayed Awwad Badawy 18P7298
Thu 25/05/2023 16:01

WrapMessage.TXT
217 bytes

Realbody Message.txt
239 bytes

2 attachments (456 bytes) Save all to OneDrive - Faculty of Engineering Ain Shams University Download all

Mail sent to verify test case 2