**DAYANANDA SAGAR UNIVERSITY**



**MINI PROJECT REPORT ON**

**SENDING MAILS FROM GMAIL USING PYTHON**

**of**

**BACHELOR OF TECHNOLOGY**

**in**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**at**

**DAYANANDA SAGAR UNIVERSITY**

**SCHOOL OF ENGINEERING, BANGALORE-560068**

**5TH SEMESTER**

**(Course Code: 16CS372)**

**OPERATING SYSTEM**

Submitted By

Shylesh S, ENG18CS0274

Shreyansh JV - ENG18CS0268

V A S Kiranmayee - ENG18CS0304

Srishti Ranjan - ENG18CS0284

***Under the supervision of***

### Prof. Gudla Balakrishna

**DAYANANDA SAGAR UNIVERSITY**

## School of Engineering, Kudlu Gate, Bangalore-560068



**CERTIFICATE**

***This is to certify that Ms. Srishti Ranjan bearing USN ENG18CS0284 , has satisfactorily completed their Operating system Mini project as prescribed by the University for the Fifth semester B.Tech. Program in Computer Science & Engineering during the year 2020 at the School of Engineering, Dayananda Sagar University, Bangalore***

Date:

Faculty in Charge

**Chairman**

**Department of Computer Science & Engineering**

**DAYANANDA SAGAR UNIVERSITY**

## School of Engineering, Kudlu Gate, Bangalore-560068



**CERTIFICATE**

***This is to certify that Ms. VAS Kiranmayee bearing USN ENG18CS0304, has satisfactorily completed their Operating system Mini project as prescribed by the University for the Fifth semester B.Tech. Program in Computer Science & Engineering during the year 2020 at the School of Engineering, Dayananda Sagar University, Bangalore***

Date:

Faculty in Charge

**Chairman**

**Department of Computer Science & Engineering**

**DAYANANDA SAGAR UNIVERSITY**

## School of Engineering, Kudlu Gate, Bangalore-560068



**CERTIFICATE**

***This is to certify that Mr. Shreyansh JV bearing USN ENG18CS0268, has satisfactorily completed their Operating system Mini project as prescribed by the University for the Fifth semester B.Tech. Program in Computer Science & Engineering during the year 2020 at the School of Engineering, Dayananda Sagar University, Bangalore***

Date:

Faculty in Charge

**Chairman**

**Department of Computer Science & Engineering**

**DAYANANDA SAGAR UNIVERSITY**

## School of Engineering, Kudlu Gate, Bangalore-560068



**CERTIFICATE**

***This is to certify that Mr. Shylesh S bearing USN ENG18CS0274 , has satisfactorily completed their Operating system Mini project as prescribed by the University for the Fifth semester B.Tech. Program in Computer Science & Engineering during the year 2020 at the School of Engineering, Dayananda Sagar University, Bangalore***

Date:

Faculty in Charge

**Chairman**

**Department of Computer Science & Engineeri**

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| **S.no** | **Contents** |
| 1 | Abstract |
| 2 | Objectives of the project |
| 3 | Requirements |
| 4 | Methodology |
| 5 | Code of the project |
| 6 | Experimental Results |
| 7 | Conclusion and future scope |
| 8 | References |

1. **ABSTRACT:**

When we were kids, we had a "Captain Midnight" decoder ring. With it, we could send messages to friends that no one else could read. Or perhaps we remember using special symbols to write notes to our friends in class.

In more serious uses, codes and ciphers are used by our military and diplomatic forces to keep confidential information from unauthorized eyes. Businesses also send data that has been encoded to try and protect trade secrets and back-room deals. After all, a person wouldn't want their competitor to know that, the person was about to acquire their company with a leveraged buy-out.

The study of enciphering and encoding (on the sending end), and deciphering and decoding (on the receiving end) is called cryptography from the Greek κρυπτός (kryptos), or hidden and γράφειν (graphia), or writing. Although the distinction is fuzzy, ciphers are different from codes. When we substitute one word for another word or sentence, like using a foreign language dictionary, we are using a code. When we mix up or substitute existing letters, we are using a cipher.

1. **OBJECTIVES OF THE PROJECT**

For a cipher to be useful, several things must be known at both the sending and receiving ends.

* The algorithm or method used to encipher the original message (known as the plaintext).
  + The key used with the algorithm to allow the plaintext to be both enciphered and deciphered.
  + The period or time during which the key is valid.

The selection of the above three items - algorithm, key and period - depend on our needs. If we are in the battlefield and are receiving current tactical data, we want an algorithm that makes it easy to decipher the message in the heat of battle. On the other hand, we must also assume that your opponent has intercepted our enciphered message and is busy trying to break it. Therefore, we must choose an algorithm (method) that is complicated enough so that by the time our opponent figures it out, the data will be worthless. The easier the algorithm we choose, the more often we will have to change the key that unlocks the code - if we want to keep your enemy in the dark.

Ciphers are broken into two main categories; substitution ciphers and transposition ciphers. Substitution ciphers replace letters in the plaintext with other letters or symbols, keeping the order in which, the symbols fall the same. Transposition ciphers keep all of the original letters intact, but, mix up their order. The resulting text of either enciphering method is called the ciphertext. Of course, we can use both methods, one after the other, to further confuse an unintended receiver as well.

1. **REQUIREMENTS**

**Hardware requirements:**

We need stable internet connection/ Wi-Fi

A device like a desktop or pc.

**Software requirements:**

Python IDE latest version

Here we are also using the MIME (Multipurpose Internet Mail Extension) module to make it more flexible. Using MIME header, we can store the sender and receiver information and some other details.

We are using Google's Gmail service to send mail. So we need some settings (if required) for Google’s security purposes. If those settings are not set up, then the following code may not work, if the google does not support the access from third-party app.

To allow the access, we need to set 'Less Secure App Access' settings in the google account. If the two-step verification is on, we cannot use the less secure access.

It uses SMTP (Simple Mail Transfer Protocol) to send the mail.

**4. METHODOLOGY**

* Create MIME
* Add sender, receiver address into the MIME
* Add the mail title into the MIME
* Attach the body into the MIME
* Start the SMTP session with valid port number with proper security features.
* Login to the system.
* Send mail and exit

1. **CODE OF YOUR PROJECT**

**PART 1:**

cryptography.fernet import Fernet

import smtplib

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

mes=input(str("Enter the message you want to encrypt :: "))

key = Fernet.generate\_key()

encoded\_message = mes.encode()

f = Fernet(key)

encrypted\_message = f.encrypt(encoded\_message)

print("The encrypted message is :: ", encrypted\_message)

mail\_body = str(encrypted\_message)

mail\_content = mail\_body

#The mail addresses and password

sender\_address = 'srishtiranjan2910@gmail.com'

sender\_pass = '\*\*\*\*\*'

receiver\_address = 'srishtiranjan29@yahoo.com'

#Setup the MIME

message = MIMEMultipart()

message['From'] = sender\_address

message['To'] = receiver\_address

#The subject line

message['Subject'] = 'The sender has sent you an encrypted message'

#The body and the attachments for the mail

message.attach(MIMEText(mail\_content, 'plain'))

#Create SMTP session for sending the mail

session = smtplib.SMTP('smtp.gmail.com', 587) #use gmail with port

session.starttls() #enable security

session.login(sender\_address, sender\_pass) #login with mail\_id and password

text = message.as\_string()

session.sendmail(sender\_address, receiver\_address, text)

session.quit()

print('Mail Sent')

**PART 2:**

o = input("Do you want to send the receiver the key? ")

if (o == 'y' or o == 'Y'):

message = MIMEMultipart()

message['From'] = sender\_address

message['To'] = receiver\_address

#The subject line

message['Subject'] = 'The sender has sent you the key'

key1 = str(key)

#The body and the attachments for the mail

message.attach(MIMEText(key1, 'plain'))

#Create SMTP session for sending the mail

session = smtplib.SMTP('smtp.gmail.com', 587) #use gmail with port

session.starttls() #enable security

session.login(sender\_address, sender\_pass) #login with mail\_id and password

text = message.as\_string()

session.sendmail(sender\_address, receiver\_address, text)

session.quit()

print('Mail Sent for the key')

**PART 3:**

op = input("Do you want to send the receiver the original text? If yes press 'y' if no press 'n' :: ")

if(op == 'y' or op == 'Y'):

message = MIMEMultipart()

message['From'] = sender\_address

message['To'] = receiver\_address

#The subject line

message['Subject'] = 'The sender has sent you the key'

d = f.decrypt(encrypted\_message)

dm = str(d)

#The body and the attachments for the mail

message.attach(MIMEText(dm, 'plain'))

#Create SMTP session for sending the mail

session = smtplib.SMTP('smtp.gmail.com', 587) #use gmail with port

session.starttls() #enable security

session.login(sender\_address, sender\_pass) #login with mail\_id and password

text = message.as\_string()

session.sendmail(sender\_address, receiver\_address, text)

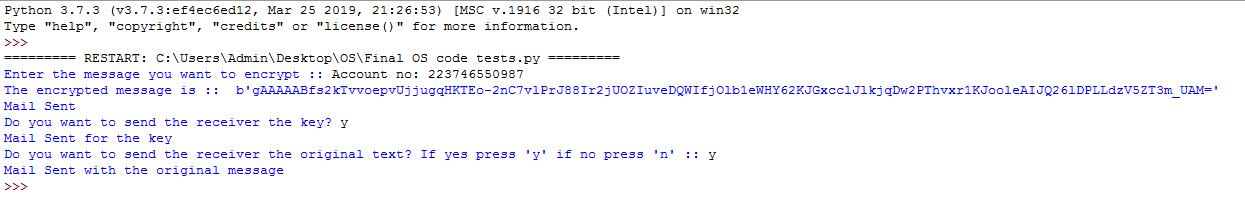
session.quit()

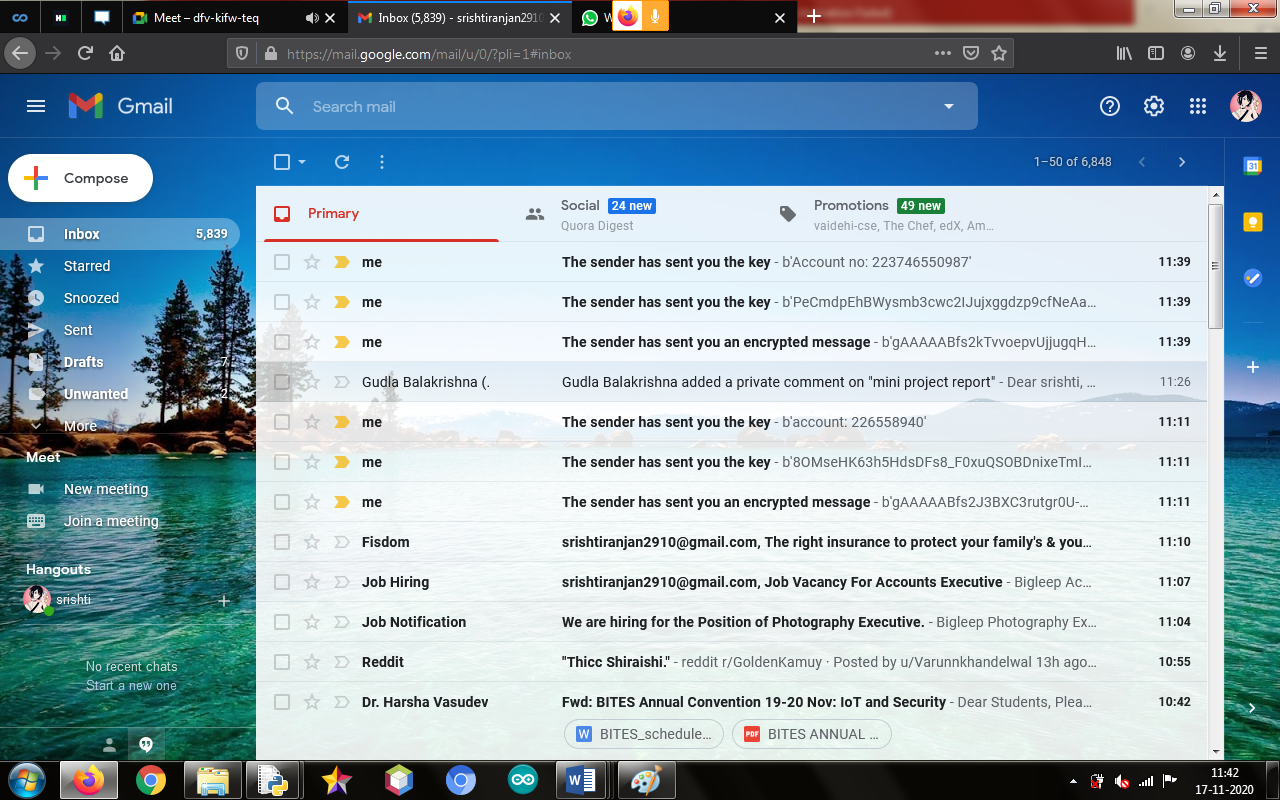
print('Mail Sent with the original message')

else:

print("The original message is not sent")

1. **EXPERIMENTAL RESULTS**

This program can be run on any python compiler, such as visual code, online python compiler and Python IDLE. The following are the screen shots of the output.****



1. **CONCLUSION AND FUTURE SCOPE**

To start with our conclusion, I’d like to reiterate the purpose of using Python to construct our emails. By using Python, we avoided sending emails manually, which automates emails in order to reduce management time and guarantees error-prone task.

Then we set up a Local SMTP Server to accomplish the similar task as we did with gmail account, which provides us with an alternative option. After setting up our platform, we dive in our email delivery with Python from bottom up.

It helps choose an algorithm (method) that is complicated enough so that by the time our opponent figures it out, the data will be worthless. The easier the algorithm we choose, the more often we will have to change the key that unlocks the code using ciphers.

In conclusion, email delivery with Python provides us with a variety of ways to construct our emails with ease. Besides, security is another concern while dealing with emails.

In the future, we could see a better implementation of security measures and we could also come up with more creative ways to compose and email with the help of pictures emails and attached files.

1. **REFERENCES**
2. <https://www.tutorialspoint.com/cryptography_with_python/cryptography_with_python_quick_guide.htm>
3. <https://www.codementor.io/python/tutorial/python-encryption-message-in-python-via-reverse-cipher>
4. <https://www.google.com/url?sa=t&source=web&rct=j&url=https://www.geeksforgeeks.org/python-message-encode-decode-using-tkinter/amp/&ved=2ahUKEwi2yequ84TtAhXOzzgGHXrkA-AQFjAFegQIJBAB&usg=AOvVaw3XlPbG84XFZgCybLnn6uR8&ampcf=1>
5. <https://www.google.com/url?sa=t&source=web&rct=j&url=https://pythonicpi.wordpress.com/2019/07/16/secret-messages-with-python/amp/&ved=2ahUKEwi2yequ84TtAhXOzzgGHXrkA-AQFjAVegQIBRAB&usg=AOvVaw0QX-ulUe6HmLjnSx8y1SCf&ampcf=1>