PROJECT REPORT on IMPLEMENATION OF SECURE EMAIL SYSTEM USING PGP AND GPG ENCRYPTIONAND DEVELOPING A PYTHON SCRIPT TO SEND ENCRYPTED EMAILS

CYBER SECURITY

Internship duration '1st June,2020' to '31st July,2020'

Submitted by:

1. Misha Dey
2. Mohd Anas
3. Jai Joshi
4. Rashmitha Jalapally
5. Divina Josy
6. Akshit Kumar
7. Ashwin Anil
8. Tanishq Mishra
9. Rohan Mittal
10. Nishant Pandey
11. Mishel Sakshi
12. Anshuman Singh
13. Thanish Vishaal

Under the Guidance and Mentorship of:

Animesh Roy

SENDING ENCRYPTED MAIL USING PGP AND EXPLORING PGP AND GPG

TASKS :

1. Sending Mail using PGP
2. Exploring PGP AND GPG

PGP :

PGP encryption or Pretty Good Privacy encryption, is a data encryption computer program that gives cryptographic privacy and authentication for online communication. It is often used to encrypt and decrypt texts, emails, and files to increase the security of emails. PGP encryption uses a mix of data compression, hashing, and public-key cryptography. It also uses symmetric and asymmetric keys to encrypt data that is transferred across networks. It combines features of private and public key cryptography. Each step uses a different algorithm, and each public key is associated with a username and an email address

PGP is a cryptographic method that lets people communicate privately online. When you send a message using PGP, the message is converted into unreadable ciphertext on your device before it passes over the Internet. Only the recipient has the key to convert the text back into the readable message on their device. PGP also authenticates the identity of the sender and verifies that the message was not tampered with in transit.

Before PGP, your Internet provider, your email provider, hackers, or the government could theoretically read your messages. PGP was developed in the 1990s to allow messages to be exchanged privately. Today, PGP has been standardized into OpenPGP, enabling anyone to write PGP software that is compatible and interoperable with other implementations.

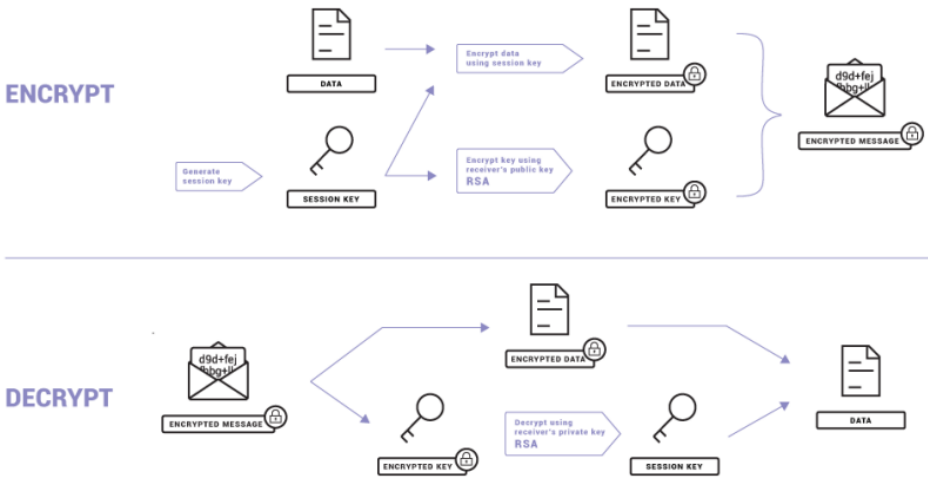
Several OpenPGP-compliant developer libraries have been created to help programmers implement PGP encryption in their applications. Proton Mail is the maintainer of two of these libraries: OpenPGP.js, for the JavaScript programming language (used in our web app), and GoOpenPGP for Go language (used in our mobile and desktop apps). OpenPGP.js, in particular, is one of the world's most widely used OpenPGP libraries and has been thoroughly audited by security experts.

Another aspect of PGP is message authentication and integrity checking. Integrity checking is used to detect if a message has been altered after it was written and to determine if it was actually sent by the claimed sender. Because the email is encrypted, changes in the message will make it unable to be decrypted with the key. PGP is used to create a digital signature for the message by computing a hash from the plaintext and producing a digital signature using the sender's private key. A person can add their signature to another person's public-key to show that it is truly that rightful owner.

PGP also ensures that the message belongs to the intended recipient. PGP includes requirements for distributing user's public keys in an identity certificate. These certificates are constructed so that tampering can be easily detected. The certificates can only prevent corruption after they have been made, but not before. PGP products also help to determine if a certificate belongs to the person that is claiming it, often referred to as a web of trust.

How does PGP work?

It's useful to see a diagram to understand how PGP encryption works. As you can see, PGP uses a combination of **symmetric key encryption** (i.e., a single-use session key encrypts and decrypts the message) and **public key encryption** (i.e., the keys unique to the recipient encrypt and decrypt the session key).



PGP combines the efficiency of symmetric encryption and the convenience of public key encryption. It is the most widely used encryption standard when it

comes to end-to-end information encryption. It is used to verify whether the sent message is genuine or not.

Difference between PGP & GPG:

PGP : Pretty Good Privacy :
PGP uses several encryption technologies, like hashing, data compression, and public/private PGP keys to protect an organization's critical information.

OPENPGP:
PGP is the backbone of Open PGP, which is an open source standard that allows PGP to be used in software that is typically free to the public.

GPG – GNU Privacy Guard:
The GPG Project provides the tools and libraries to allows users to interface with a GUI or command line to integrate encryption with emails and operating systems like Linux.
GPG can open and decrypt files encrypted by PGP or Open PGP, meaning it works well with other products.

- PGP--proprietary solution owned by Symantec
- GPG--an open source standard

TOOL : OPENPGP : (Open Specification For pretty good privacy)

Projects that use openpgp.js library:
Proton Mail
Mailvelope

https://github.com/openpgpjs/openpgpjs/blob/master/README.md#getting-started
https://github.com/openpgpjs/openpgpjs#platform-support

How to install and configure openpgp in Linux?
https://hostadvice.com/how-to/how-to-install-and-configure-openpgp-on-ubuntu-18-04/

TOOL : GNUPGP

The GNU Privacy Guard. GnuPG is a complete and free implementation of the OpenPGP standard as defined by RFC4880
(also known as PGP). GnuPG allows you to encrypt and sign your data and communications; it features a versatile key

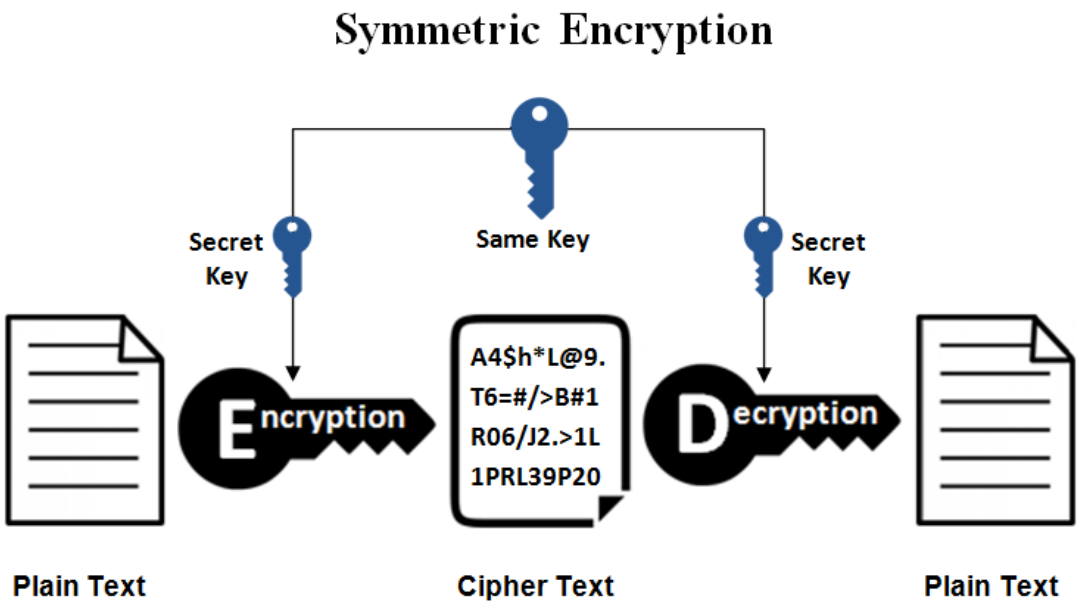
management system, along with access modules for all kinds of public key directories.

Sending Mail using PGP Encryption:

❖ Type of keys:

Symmetric key- It is an encryption that involves only one secret key(either be a number, a word or a string of random letters) to cipher and decipher information. The sender and the recipient should know the secret key that is used to encrypt and decrypt all the messages.. The most widely used symmetric algorithm is AES-128, AES-192, and AES-256.

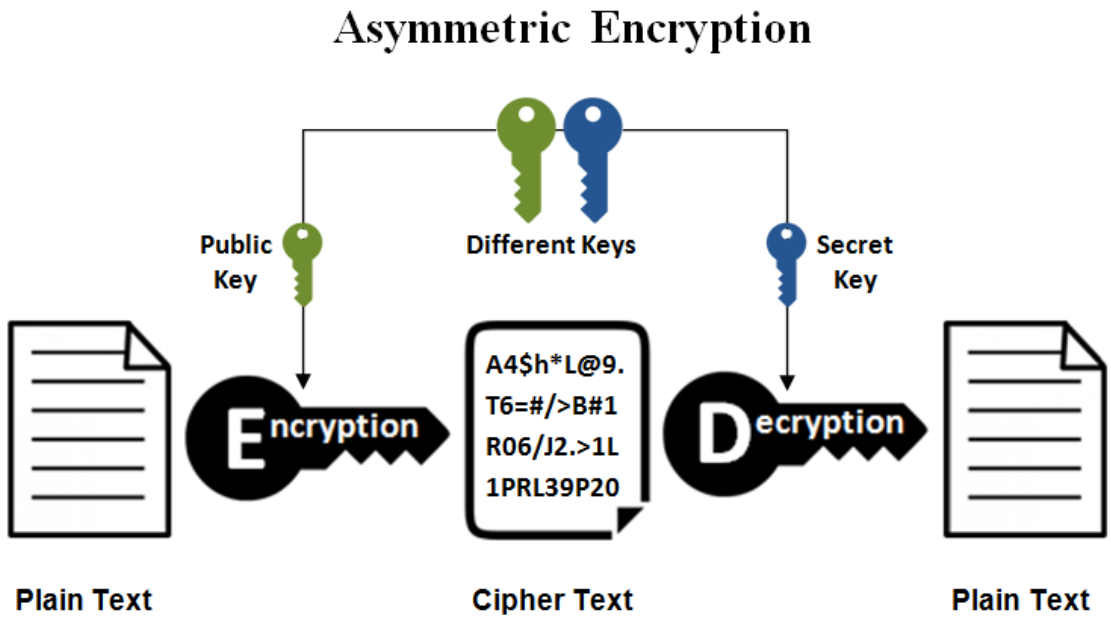
The main disadvantage of the symmetric key encryption is that all parties involved have to exchange the key used to encrypt the data before they can decrypt it.



Asymmetric key - It is an encryption (public key cryptography), uses two keys to encrypt a plain text. It ensures that malicious persons do not misuse the keys and this is why it uses two keys to boost the security. A public key is made freely available to anyone who might want to send you a message. The second private key is kept a secret so that you can only know.

A message that is encrypted using a public key can only be decrypted using a private key, while also, a message encrypted using a private key can be decrypted using a public key. Security of the public key is not required because it is publicly available and can be passed over the internet.

This has a far better power in ensuring the security of information transmitted during communication.Popular asymmetric key encryption algorithm includes EIGamal, RSA, DSA, Elliptic curve techniques, PKCS.



LIBRRIES/TOOLS/links REQUIRED :

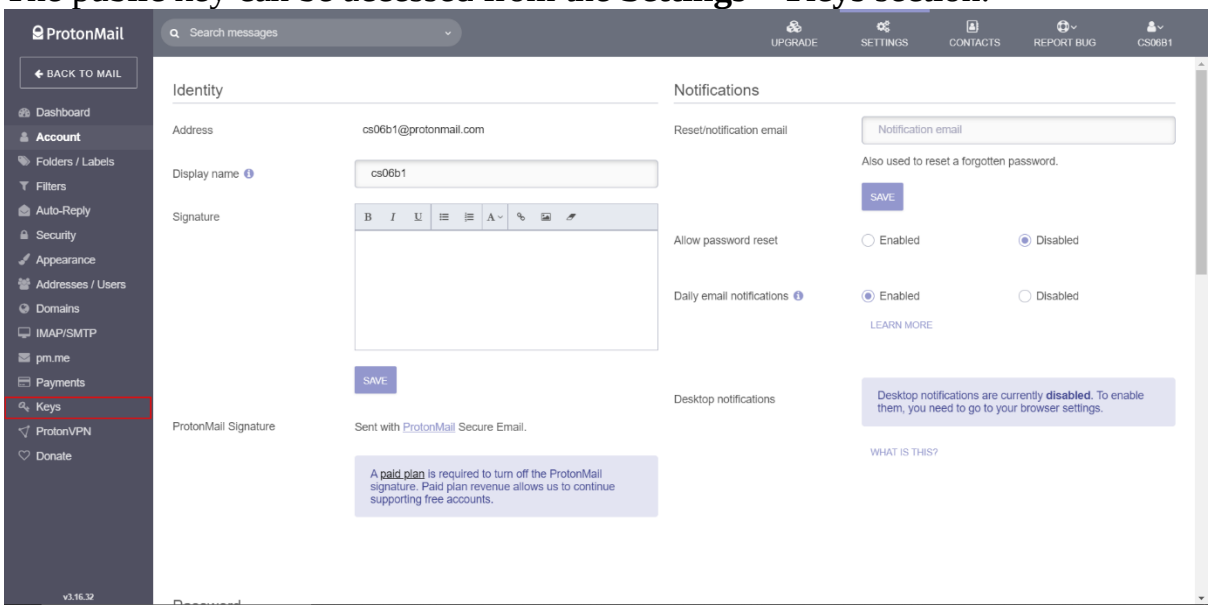
1. OPENPGP
2. GPA
3. PYCRYPTODOMEX
4. GNUPG
5. GPG
6. Protonmail

7. links:
    1. https://www.guerrillamail.com/inbox
    2. https://temp-mail.org/

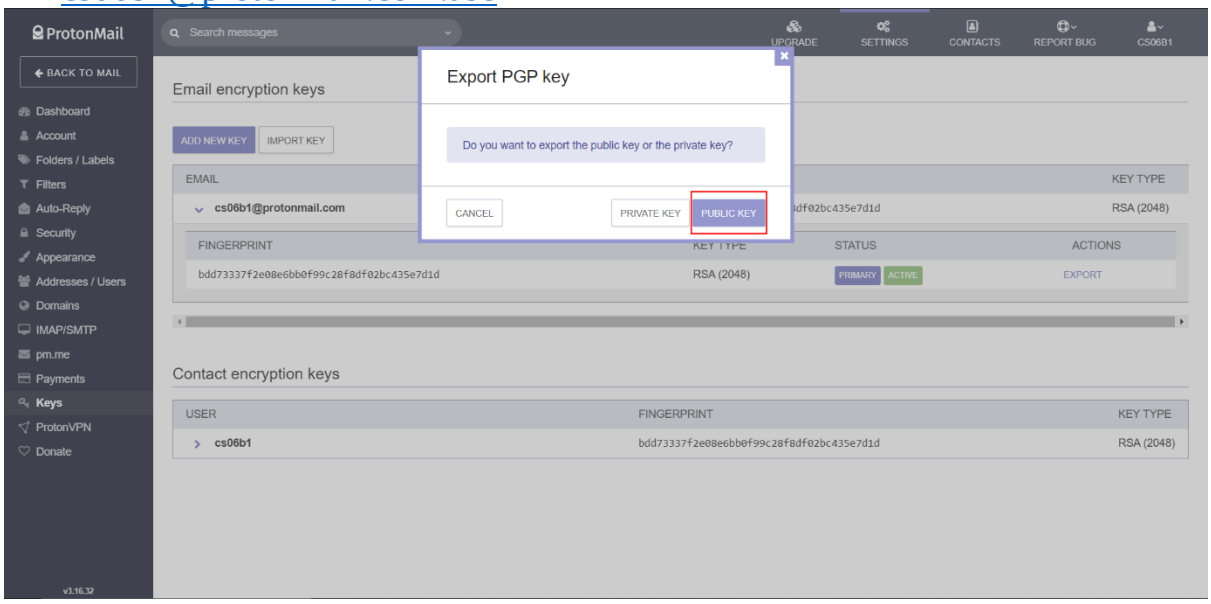
Steps:

Step-1:
Getting recipient's public key

The public key can be accessed from the **Settings > Keys** section.



Now, to download the public key, just click on **EXPORT** in the **Actions** Column and tap on **PUBLIC KEY.** A .asc file will be downloaded. In this particular case it's cs06b1@protonmail.com.asc



Step-2:
Importing Recipient's public key

To import recipient's public key to pgp, just type this card:

gpg --import publickey.cs06b1@protonmail.com.asc

Output:

```
gpg: key F8DF02BC435E7D1D: public key "cs06b1@protonmail.com <cs06b1@protonmail.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1
```

To list keys available:

gpg –list-keys

Output:

```
/root/.gnupg/pubring.kbx
------------------------
pub   rsa2048 2020-08-04 [SC]
      BDD73337F2E08E6BB0F99C28F8DF02BC435E7D1D
uid           [ unknown] cs06b1@protonmail.com <cs06b1@protonmail.com>
sub   rsa2048 2020-08-04 [E]
```

Step-3:

Crafting a message and encrypting the same with the imported public key

Now we just need to compose a message, save it as a text file and encrypt it with the public key of the recipient.

Composed message:

```
root@kali:/home/frost/Desktop# cat secret-message.txt
 "noine noine" not "nine nine"



Regards
Jake
```

In this case, composed message is saved as secret-message.txt

To encrypt:

gpg --encrypt –armor –r cs06b1@protonmail.com secret-message.txt

Output:

```
gpg: 4CD82CA6FEBC4207: There is no assurance this key belongs to the named user

sub  rsa2048/4CD82CA6FEBC4207 2020-08-04 cs06b1@protonmail.com <cs06b1@protonmail.com>
 Primary key fingerprint: BDD7 3337 F2E0 8E6B B0F9  9C28 F8DF 02BC 435E 7D1D
      Subkey fingerprint: 2155 8635 5023 82FD 9C90  4A64 4CD8 2CA6 FEBC 4207

It is NOT certain that the key belongs to the person named
in the user ID.  If you *really* know what you are doing,
you may answer the next question with yes.

Use this key anyway? (y/N) y
```

Let's take a look at the encrypted message:

cat secret-message.txt.asc

Output:

```
-----BEGIN PGP MESSAGE-----

hQEMA0zYLKb+vEIHAQf/ZtZ1e3wKSSZu8WcuL8NBpobdxhLpbTOi2+n2EWkwjA0V
WObZ1FNPec3MdZlqMjHOF6JaQ+wQwTEfOMmCduxnoHjm0cvV9lgCVqpNOUd0b3qn
3bKXI38iB5QtG6p99WDnGHnJd7u6ctsirGQIcLWeHEHHRbPQ4F4OuhC7Hhpq97Bn
vKu5m8E5YiaTgbU7vIrT50oeJ4WwilbVS1QsUdLn9L1gaiHniSVe1VDZkeEY4H81
Jn+mact+phR86qHZPIIkH1UOUPUzdnjk2eKF0ptG11PcrByj1CB2Z88IPTrBnZeo
M2fgUKE3SYD8bZDJ80V+4LxkINKgLlovGM64DmWfKNJvAQkzNVuKzLeeB1UAIclB
qsBlc/PIPRljtJX/Lhj+IpLfyRS30vGJS+rUWTL60LjCcCRESu0w9kxt7TUXK521
Qh8A+ZTzMuuSLFdZp+PPXD/7WEdkfhBHSqA72257W2mlacs+1NAaFyZhPwao9Jqc
=/JiR
-----END PGP MESSAGE-----
```

Step-4:

Sending the encrypted message via a disposable mail service

For this step, we are going to use:
https://www.guerrillamail.com/

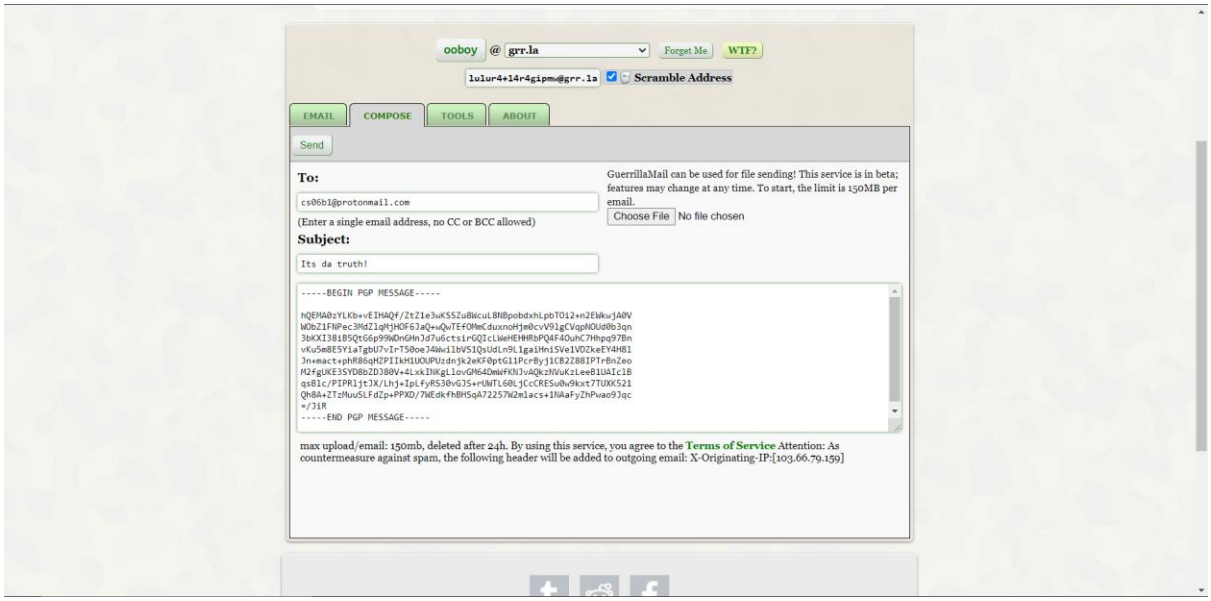
Composing the email:

We just need to copy the contents of the encrypted message .asc file:

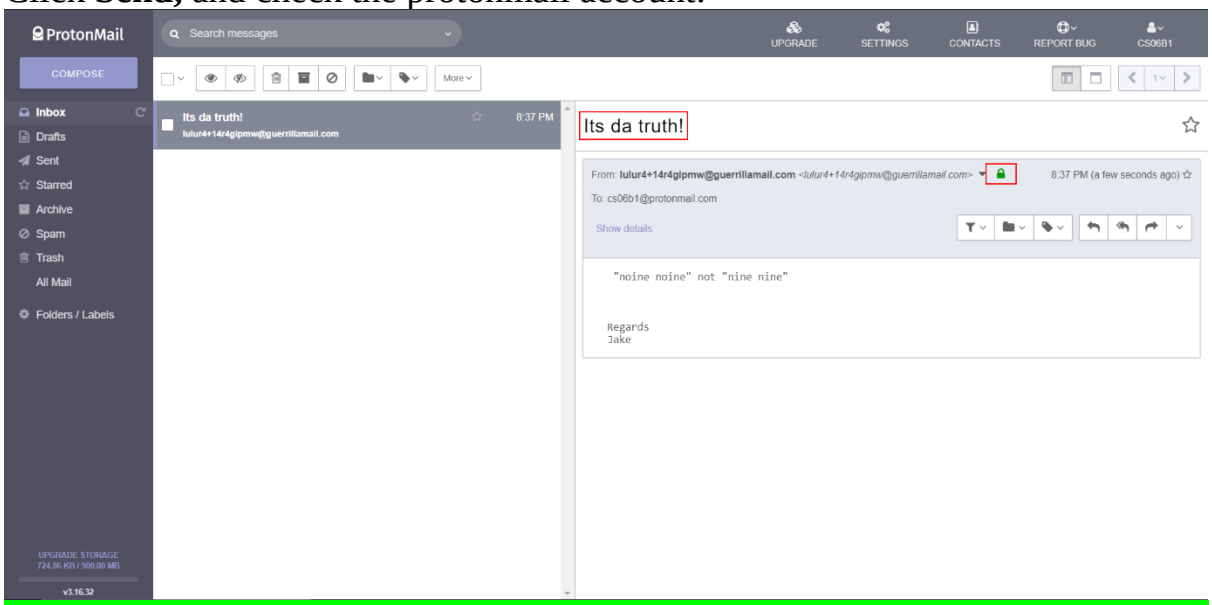
secret-message.txt.asc

Recipient Mail Address : cs06b1@protonmail.com

Subject : Its da truth



Click **Send,** and check the protonmail account:



The green padlock sign aside the sender's email verifies that this message came via end-to-end encryption.

To verify:

The contents and the subject of the mail composed is accurately the same.

Principles of secure email system through PGP

It provides

Authentication through the use of digital signature,

The confidentiality through the use of symmetric block encryption,

Compression using the ZIP algorithm, and

E-Mail compatibility using the radix-64 encoding scheme.

To Generate Keys on the PGP Command Line:

Use the --gen-key command to create a new key pair.

The --gen-key command automatically creates your key pair and a public and a private keyring in the home directory,

pgp --gen-key <user> --key-type <type> --encryption-bits <bits> --passphrase <pass> [--signing-bits <bits>] [options]

Example:

pgp --gen-key "Alice Cameron <alice@example.com>" --key-type rsa --encryption-bits 2048 --signing-bits 2048 --passphrase cam3r0n --expiration-date 2007-06-01

<user> is a user ID that people can use to locate your public key. A common user ID is your name and email address in the format: "Alice Cameron <alice@example.com>". If your user ID contains spaces, you must enclose it in quotation marks.

<type> means you are creating either an RSA or a DH key.<bits> is the number of bits of the key (usually 1024 - 4096).

<passphrase> is a passphrase of your choice. If your passphrase includes spaces, enclose it in quotation marks.

NOTE: You can locate your keyrings using the --version -v command.

Exporting Your Public Key to a Text File:

The command --export exports only public keys, while the command --export-key-pair exports private keys.

pgp --export/--export-key-pair <input> [options]
where:
– <input> is the user ID, portion of the user ID, or the key ID of the key you want to export.

[options] change the behavior of the command. Options are:
--output lets you specify a different name for the exported file.

If you don't enter any input, all keys on the keyring are exported.

By default, keys are exported as ASCII armor (.asc) files into the directory currently active on the command line.

Example:
pgp --export example

All keys with the string “example” anywhere in them would be exported into separate .asc files.

pgp --export “Alice C <acameron@example.com>”
Only keys that exactly match this user ID would be exported. The filename would be Alice C.asc.

Importing a Public Key:

pgp --import <input> [<input2> ...] [options]

Encrypt & Sign a file:

pgp --encrypt report.txtExample -recepient public key --sig report.txtExample --signer "ur keyid" --passphrase <abc>

Signing only:

There are three main options to perform signing in PGP commandline --sign, --clearsign, and --detached. These options are very different from one another and they each have their own use cases.

- **--sign** is used to sign all file types including binary-based files. When using the --sign option remember to include the .pgp file extension so the file can be decrypted as all signed files are encapsulated in the signed file. Using the decrypt option would be used to verify the signed file. the --decrypt option can be used without putting in a passphrase.

```
pgp --sign report.txt
```

```
Example :     --signer "the signing key" --passphrase "your passphrase here"
```

- **--clearsign** is only used for regular text documents such as notepad or ASCII format. The --clearsign option cannot be used with non-text file format. For example signing an Excel spreadsheet would result in a courrupted file that can no longer be used.

```
pgp --clearsign report.txt
```

```
Example:  --signer "the signing key" --passphrase "your passphrase here"
```

- **--detached** will output a single .sig so both the original file and the .sig file will be needed to verify the signature. This signing option can be used with all file types.

```
pgp --detached report.txt
```

Example: --signer "the signing key" --passphrase "your passphrase here"

Decrypt a file:

pgp --decrypt <input> [<input2> ...] [<inputd>...] [options]

Example:
pgp --decrypt --input "D:\Folder\h837.20120613.13996.pgp" --passphrase "Passphrase Removed"

Conclusions:

- Sensitive information is always protected. It cannot be stolen or viewed by others on the internet. It assures that the information that is sent or received was not modified in transmission and that files were not changed without your knowledge.
- Information can be shared securely with others including groups of users and entire departments.
- You can be certain who the email is from and who it is for. PGP verifies the sender of the information to ensure that the email was not intercepted by a third party.
- Your secure emails and messages cannot be penetrated by hackers or infected by email attacks.
- Others cannot recover sensitive messages or files once you have deleted them.
- PGP encryption software is very easy to learn how to use. With virtually no training, users are able to learn how to use it right away.

PYTHON SCRIPT TO SEND EMAIL USING SMTP

Sending a encrypted email through python can be done using email package (library for managing email messages) and smtplib (more details below).

LIBRARIES,MODULES AND PACKAGES USED :

1.smtplib - SMTP Protocol Client :

Smtplib ,an library or a module that defines an SMTP(Simple Mail Transfer protocol)client session object which can be used to send any email to any other internet machine with an SMTP or ESMTP listener daemon.

2. ssl - Secure Socket Layer :

Secure Socket Layer (SSL) provide security to the data that is transferred between web browser and server. SSL encrypt the link between a web server and a browser which ensures that all data passed between them remain private and free from attack.

3.Email Package : The email package is a library for managing email messages, including MIME and other RFC 2822-based message documents. It is specifically not designed to do any sending of email messages to SMTP (RFC 2821), NNTP, or other servers; those are functions of modules such as smtplib and nntplib.

4.MIME - Multipurpose Internet Mail Extensions :

MIME(Multipurpose Internet Mail Extensions) is an Internet standard that extends the format of email messages to support text in character sets other than ASCII, as well as attachments of audio, video, images, and application programs.

MimeMultipart type is one which represents a document that's comprised of multiple component parts, each of which may have its own individual MIME type.If a message has a multipart Content-Type, that means it consists of multiple messages and each of them defines its own Content-Type (which can again be multipart or something else).

MimeBase is the base class for all the MIME-specific subclasses of Message.

MIMEText class is used to create MIME objects of major type text. The email package provides some convenient encodings in its encoders module. These encoders are actually used by the MIMEAudio and MIMEImage class constructors to provide default encodings.

5. getpass - Portable Password input :

The getpass module provides a portable way to handle such password prompts securely.

In [12]:

```
import os
import ssl
import smtplib
import getpass
import pandas as pd
from email.mime.multipart import MIMEMultipart
from email.mime.multipart import MIMEBase
from email.mime.text import MIMEText
from email import encoders
```

To create the message template, the external file path is used as argument to store message body

In [13]:

```
def create_message_template(ext_message_file_path):
    message_template_file=open(ext_message_file_path,mode='w',encoding='utf-8')
    message_template_file.write(input("Enter your message:"))
```

```
    message_template_file.close()
```

The recipent details (recipents list) will be stored in external file path provided

In [14]:

```
def create_recipents_list(ext_contacts_file_path):
    recipents=open(ext_contacts_file_path,mode='w',encoding='utf-8')
    rescipent_no=input("Enter the no. of rescipents:")
    for i in range(rescipent_no):
        recipent_list=input("Enter your Recipents address: ")
        recipents.write(recipents_list+'\n')
    recipents.close()
```

To extract the list of recipents from external file,function extract_recipents_ext_file takes a ext_contacts_file_path as its argument. The data is split it into two lists recipents_name and email_id

In [15]:

```
def extract_recipents_ext_file(ext_contacts_file_path):
    recipent_name=[]
    email_id=[]
    with open(ext_contacts_file_path, mode='r' , encoding ='utf-8') as ext_cont_file:
        for contact in ext_cont_file:
            recipent_name.append(contact.split()[0])
            email_id.append(contact.split()[1])
    return recipent_name,email_id
```

The function extract_message_template extracts message from the external file path provided as argument as an object

In [16]:

```
def extract_message_template(ext_message_file_path):
    with open(ext_message_file_path,'r',encoding='utf-8') as message_template_file:
        extracted_message=message_template_file.read()
    return extracted_message
```

To personalize the email for the recipent create the MIMEMultipart message object and load it with appropriate headers for From, To, and Subject fields.And then attach the message body to the mail.

A MIME attachment with the content type "application/octet-stream" is a binary file. Typically, it will be an application or a document that must be opened in an application, such as a spreadsheet or word processor.

The payload is the part of transmitted data that is the actual intended message. set_payload((attachment).read()) , Sets the entire message object's payload to payload.

Encoders Encodes the payload into base64 form and sets the Content-Transfer-Encoding header to base64. This is a good encoding to use when most of your payload is unprintable data since it is a more compact form than quoted-printable.

add_header extended header setting that adds a header to the message , additional header parameters can be provided as keyword arguments.

Content-Disposition is the header field to add and attachment is the primary value for the header. And attach the MIME with the "application/octet-stream" to the message. Finally the personalized email is returned.

In [17]:

```
def send_personalized_email(user_Address,recipent_mail,extracted_message,subject,file_to_
attach,attachment,filename):
    message=MIMEMultipart()
    message['From']=user_Address
    message['To']=recipent_mail
    message['Subject']=subject
    message_body=MIMEText(extracted_message,'plain')
```

```
    message.attach(message_body)
    attach_f=MIMEBase('application','octet-stream')
    attach_f.set_payload((attachment).read())
    encoders.encode_base64(attach_f)
    attach_f.add_header('Content-Disposition',"attachment; filename =%s" % filename)
    message.attach(attach_f)
    return message
```

Import the smtplib and then create an SMTP instance that encapsulates an SMTP connection. It takes as parameter the host address and a port number, both of which entirely depends on the SMPT server settings of your particular email service provider.

smtplib.SMTP(smtp_server, port) is used to create an SMTP object.

Extended HELO (EHLO) is an Extended Simple Mail Transfer Protocol (ESMTP) command sent by an email server to identify itself when connecting to another email server to start the process of sending an email.To identify yourself to the server, .ehlo() should be called after creating an .SMTP() object, and again after .starttls().

Instead of using .SMTP_SSL to create a connection that is secure from the outset, we can create an unsecured SMTP connection and encrypt it using .starttls().

For each recipents_name and email_id in zip(recipent_name,res_email_id) send_personalized_email function is called to send personalized email to each recipent email_id.

In [18]:

```
def set_SMPT_server(smtp_server,port,user_Address,User_Password,ext_contacts_file_path,ext_message_file_path):
    subject=input("Enter Subject: ")
    #create_recipents_list(ext_contacts_file_path)
    create_message_template(ext_message_file_path)
    extracted_message=extract_message_template(ext_message_file_path)
    print(extracted_message)
    recipent_name,res_email_ids=extract_recipents_ext_file(ext_contacts_file_path)
    print(recipent_name,res_email_ids)
    context = ssl.create_default_context()
    file_to_attach=input("Enter the complete file path you wish to attach :")
    attachment=open(file_to_attach,"rb")
    filename=input("Rename the File: ")
    with smtplib.SMTP(smtp_server, port) as server:
        #server=server.smtplib.SMTP(host_Address,port)
        server.ehlo()
        server.starttls()
        server.ehlo()
        server.login(user_Address,User_Password)
        print("Congratulations!! Connection established!")
        recipent_details=zip(recipent_name,res_email_ids)
        for recipent_name,recipent_mail in recipent_details:
            message=send_personalized_email(user_Address,recipent_mail,extracted_message,subject,file_to_attach,attachment,filename)
            server.send_message(message)
            print("MESSAGE DELIVERED to {} !!".format(recipent_mail))
        server.quit()
```

Getting the working directory path and the credentials and passing the details as arguments to the set_SMPT_server function.SMTP host address(for gmail) is 'smtp.gmail.com'. Port number used for gmail is '587'.

In [19]:

```
def main():
    directory_base_path=input("Enter the working directory path:")
    ext_contacts_file_name=input("Enter the filename containing recipents list:")
    ext_contacts_file_path=os.path.join(directory_base_path,ext_contacts_file_name)
    ext_message_file_name=input("Enter the filename containing the message :")
    ext_message_file_path=os.path.join(directory_base_path,ext_message_file_name)
    host_Address=input("Enter the smtp server Address:")
    port=int(input("Enter the port number:"))
```

```
    user_Address=input("Enter User Email Address:")
    print("Enter User Password: ")
    User_Password=getpass.getpass()
    set_SMPT_server(host_Address,port,user_Address,User_Password,ext_contacts_file_path,
ext_message_file_path)
```

In [20]:

```
if __name__ == '__main__':
    try:
        main()
        print("THANK YOU. \n")
    except:
         print("Sorry!! Failed to send email!!")
```

```
Enter the working directory path:/home/mishadey/Desktop/MAJOR_PROJECT/CS_MAJOR_DRAFT
Enter the filename containing recipents list:ext_rescipents_file.txt
Enter the filename containing the message :message_template.txt
Enter the smtp server Address:smtp.gmail.com
Enter the port number:587
Enter User Email Address:misha.2june@gmail.com
Enter User Password:
........
Enter Subject: CYBER SECURITY MAJOR PROJECT TEST MAIL
Enter your message:THIS IS THE FINAL TEST MAIL.JUST CONFIRM IF YOU GET IT.
THIS IS THE FINAL TEST MAIL.JUST CONFIRM IF YOU GET IT.
['mishadey', 'mishelsakshi', 'ashwinrockonn123', 'rashmijalapally', 'singh.anshuman.singh
8', 'tanishqm033', 'thanishvishal', 'theakshitkumar', 'uchiha72000', 'jaijoshi0310', 'mit
talrohan2001', 'mohdanas1612', 'pandeynishantsagar'] ['misha.2june@gmail.com', 'mishelsak
shi@gmail.com', 'ashwinrockon123@gmail.com', 'rashmijalapally@gmail.com', 'singh.anshuman
.singh8@gmail.com', 'tanishqm033@gmail.com', 'thanishvishal@gmail.com', 'theakshitkumar@g
mail.com', 'uchiha72000@gmail.com', 'jaijoshi0310@gmail.com', 'mittalrohan2001@gmail.com'
, 'mohdanas1612@gmail.com', 'pandeynishantsagar@gmail.com']
Enter the complete file path you wish to attach :/home/mishadey/Desktop/MAJOR_PROJECT/CS_
MAJOR_DRAFT/Verzeo_Major.pdf
Rename the File: Verzeo_Major.pdf
Congratulations!! Connection established!
MESSAGE DELIVERED to misha.2june@gmail.com !!
MESSAGE DELIVERED to mishelsakshi@gmail.com !!
MESSAGE DELIVERED to ashwinrockon123@gmail.com !!
MESSAGE DELIVERED to rashmijalapally@gmail.com !!
MESSAGE DELIVERED to singh.anshuman.singh8@gmail.com !!
MESSAGE DELIVERED to tanishqm033@gmail.com !!
MESSAGE DELIVERED to thanishvishal@gmail.com !!
MESSAGE DELIVERED to theakshitkumar@gmail.com !!
MESSAGE DELIVERED to uchiha72000@gmail.com !!
MESSAGE DELIVERED to jaijoshi0310@gmail.com !!
MESSAGE DELIVERED to mittalrohan2001@gmail.com !!
MESSAGE DELIVERED to mohdanas1612@gmail.com !!
MESSAGE DELIVERED to pandeynishantsagar@gmail.com !!
THANK YOU.
```

In [ ]:

```
```

ext_contacts_file content:

mishadey misha.2june@gmail.com
mishelsakshi mishelsakshi@gmail.com
ashwinrockonn123 ashwinrockon123@gmail.com
rashmijalapally rashmijalapally@gmail.com
singh.anshuman.singh8 singh.anshuman.singh8@gmail.com
tanishqm033 tanishqm033@gmail.com
thanishvishal thanishvishal@gmail.com
theakshitkumar theakshitkumar@gmail.com
uchiha72000 uchiha72000@gmail.com
jaijoshi0310 jaijoshi0310@gmail.com
mittalrohan2001 mittalrohan2001@gmail.com
mohdanas1612 mohdanas1612@gmail.com
pandeynishantsagar pandeynishantsagar@gmail.com

message_template:

Dear Sir/Ma'am,

This is a test mail from group CS06B1.

Kindly, accept it.

Thank You.

DISCUSSION

Q1.What can you do to ensure your privacy during sending email?
Ans:1. Use two-factor authentication

The basic principle of two-factor authentication is simple: combine something you know with something you have. One example is a debit card, which requires you to have both your physical card and your PIN to verify your identity. By enabling two-factor authentication (or two-step verification), you aren't putting all of your faith in a password. That's a good thing, considering how weak many of our passwords are. For Gmail, setting up two-step verification is as simple as clicking a button and entering in your mobile number. For Windows Mail, or Outlook, it's a similar process. Just log in, go to your "Password and security" tab and click "Set up two-step verification." Now that you've enabled two-factor authentication, a hacker with your password is out of luck — unless they've also managed to steal your cell phone.
2. Limit forwarding

When we're sent a message we want to share, we often click "Forward" without thinking about the consequences. Where is the message going? Who will see it? Where will it be stored? If your email is hosted on a corporate server, it is likely there are certain security measures in place to protect any sensitive information contained in your private email. When someone forwards an internal email to a recipient outside of your company, however, you are exposing that data (as well as any other emails in the forwarded chain) to potentially unsecured, unencrypted servers.Similarly, if you're a covered entity sending email containing protected health information (PHI) to a business associate, all it takes is one employee to forward that email to an unauthorized recipient to violate HIPAA.
3. Set expiration dates on your messages

While some of us can't stand a messy inbox, the average user doesn't bother cleaning up their private email, often seeing deleting email as a waste of time. Considering more than 50 percent of us receive at least 11 emails a day, can you blame them?That means that any sensitive information you send to a client could very well be sitting there months later. At that point, you no longer control the fate of your data.Luckily, Virtru lets you set an expiration date on your email, so that after a certain date, it will no longer be readable by the recipient (or anyone else, for that matter).
4. Understand your service provider's TOS

Your email provider's terms of service can tell you a lot more than their media interviews and advertisements can. For starters, it'll let you know what kind of security they are offering you. Are they encrypting messages on their server? Do they have protections against brute-force attacks? Is there any guarantee that your data is being protected? While you might think your email provider has your best interests in mind, there's a good chance that they don't have the same expectations you do. Take Google for example, which openly passes private email through automated scanning. After reading your email provider's TOS, you'll likely realize that keeping your private email secure isn't their first priority — that's entirely up to you.
5. Encrypt your email

The best way to keep your private email away from prying eyes and hackers is to use encryption. Encryption protects your private email by jumbling up your messages, making them impossible to decipher unless you explicitly authorize someone to read them.If you are using a client-side encryption service like Virtru, even if your inbox is compromised, the contents of your message will be unreadable. Likewise, you don't have to worry about your messages being intercepted after you send them, either by hackers or nosey service

providers. As an added bonus, if your email ends up getting stored on a server outside of your control, you still have power over who gets to see it — and you can revoke that permission at any time.

While email may not have been designed to be secure, but users can enjoy added privacy and security with a few workarounds.

Q2.What privacy should you expect while sending email?
Ans:The proliferation of high-speed communication devices have made us more productive and more vulnerable in terms of our privacy. Desk-bound workers may be tempted to use the office email account to engage in personal communications – however, they do so at some risk to their privacy. How much privacy can employees expect for their electronic communications at work? Practically speaking, it is safe to presume everything may be monitored by your employer.

The law generally favors employers' interests over employees' privacy. Employers clearly have a legitimate business interest in tracking employee time and productivity. Additionally, employers must ensure their workers are not engaging in any illegal activity or releasing trade secrets. The law permits employers to read employee email messages; if there is a company policy in place that assures employees that email messages will remain private, a worker may be able to argue that there was a reasonable expectation of privacy, but the effectiveness of that argument varies. The courts have generally upheld employers' rights to monitor and read their employees' email messages, particularly when there is a compelling, business-related reason for doing so.

There is, however, a law that affords employees some protection of their privacy when accessing personal email accounts, such as Gmail or Hotmail. Should the password to a personal email account fall into the wrong hands, the employer is prohibited from using that information to access the employee's personal emails without the employee's permission. Under the Stored Communications Act, such conduct is a crime and also creates a civil cause of action for damages. Keep in mind that accessing these accounts from a work computer can give the employer the right to read messages employees send or receive using company equipment. However, the employer is not permitted to log in and view other personal email communications.

Q3.How to send a secret message?
Ans:We can use disposable emails from the sites such as gueriilamail.com and use them effectivelySent emails aren't always secure. But if you need to send a private email to someone, you can create a fake email account. Share the login and password with the person who is receiving the message. Compose an email message, but save it in the drafts folder instead of sending it.

Q4.How can you automate email?
Ans:1 — Use Labels to Automatically Sort Incoming Messages

You are receiving tons of emails and they all have different level of importance. To manage them properly, you can create several categories and then sort messages manually, but it can take a lot of time. Luckily, there is a way automatically sort emails into folders in Gmail.

Right in the search bar, there is a dropdown menu that allows you to sort messages by subject, sender, or date. What is more, you can also set up certain terms as labels for more specific filtering. Gmail can automatically label incoming messages so you can just click on this mark and read all related emails.
Moreover, you're free to set up as many labels as you need to organize your inbox as precisely as possible.

Step-by-step guide:

Click on the dropdown arrow next to the search bar;

Type the label word into the subject line and use "Create filter" command;

Check the "Apply the label:" box, open the dropdown menu and click "New label…";
Type in your label, click "Create" and "Create filter";
Now you can see the label next to all incoming messages with subjects that contain the respective search criteria.

2 — Use Filters to Automatically Forward and Mark Emails

Another trick to keep your inbox organized is to deal with irrelevant messages or requests that can be completed by your assistants. For example, if someone sends you an email and mentions a meeting in a subject line, you might want to forward it to your secretary and don't worry about scheduling.
You can sort Gmail emails by sender or subject and automatically forward them to your assistant or relevant department. You can also mark emails as read or important, move them to another folder, etc.

Step-by-step guide:

At the top of the screen find the search bar and click the dropdown button;
Type in the keyword in the "Subject" field and use "Create filter" command;
Find the "Forward it" box and fill in the address of your choice;
Also, you can click "Mark as read" so the messages won't be displayed as unread;

Now all emails with the specific keyword in the subject line will be automatically sent to the chosen address.

Important:
Note that you would first need to set the forwarding addresses in the "Forwarding and POP/IMAP" section of Gmail settings.
3 — Use Canned Responses to Send Pre-Written/Saved Responses

If you have to respond to numerous typical requests or send multiple identical follow-ups to your clients, you probably write almost the same messages. So, why not create several templates and use them to answer certain types of emails? Gmail offers you a possibility to store such saved responses in your mailbox and reply to repetitive emails without ever having to write the same letters.
If you want to use pre-written emails, Gmail's canned responses is the tool you need. You can enable it in "Settings" (the gear icon) and save any message as a canned response. Use such automatic emails in Gmail to quickly answer incoming requests just in a few clicks.

Step-by-step guide:

Open Gmail settings, find “Advanced” tab and switch the “Canned Responses” to “Enable”. Save the changes and return to the main Gmail screen;
Write a message in the Compose window, find the dropdown menu in the lower right-hand corner of the screen, and click “Canned responses”;
Choose “New canned response”, name it and save the changes;
When you receive an email, click “Reply” and open the three vertical dots dropdown menu. Head to “Canned responses” and choose the template you want to insert.

4 — Remove Promotional Emails out of Your Inbox Automatically

Marketing emails, newsletters, and automatic responses from websites can indeed overload your inbox. They don’t require much time to handle but still can distract you from more important messages. So, if you don’t want to lose focus every time you receive such messages, get rid of them as soon as they appear in your inbox.
The easiest way to get out of annoying promotions is to use Gmail filters. You can select the message you have received and specify to filter similar emails from this address. After sorting emails by sender, you can choose what to do with them: label, archive, or delete. Repeat this for every incoming marketing email to ensure that you won’t receive them in the future.

Step-by-step guide:

Open the promotional email and click on the drop-down menu at the top-right;
Gmail will automatically fill the sender details in the “From” field, so just choose “Filter messages like this”, and then “Create filter ” in the popup window;

In the next window you can decide what to do with such messages;

Now all emails that match criteria, will be automatically removed from your inbox.

Helpful Scripts to Automate Your Gmail

The listed methods work great for most repetitive tasks, but there are some other ways to organize your mailbox. If you want to know how to automate emails in Gmail even more, there are some scripts that can help with more advanced tasks.
Auto delete emails after X number of days

We usually keep emails in our inbox even if they are not useful. To keep your message history clean, you might want to remove certain emails after several days. Luckily, you can do it using Google Apps Script, the powerful tool for G-Suite app automation. In this program, select “Blank Project” and paste the following script:

```
function auto_delete_mails() { var label = GmailApp.getUserLabelByName("Delete Me"); // A label that signifies emails marked for deletion if(label == null){ GmailApp.createLabel('Delete Me'); } else{ var delayDays = 2 // Number of days before messages are moved to trash var maxDate = new Date(); maxDate.setDate(maxDate.getDate()-delayDays); var threads = label.getThreads(); for (var i = 0; i < threads.length; i++) { if (threads[i].getLastMessageDate()<maxDate){ threads[i].moveToTrash(); } } } }
```

Define the delay before the email will be deleted (delayDays), save the script and set a trigger to run it daily. This script will create a label “Delete Me” that you can assign to useless

messages. You can even categorize incoming mail so that different letter types would be deleted in different terms.

Snooze your emails

Sometimes you might need to return to messages several days after you read them. To do this, you can create another Google script with such code:

```
var MARK_UNREAD = false; var ADD_UNSNOOZED_LABEL = false; function getLabelName(i) { return "Snooze/Snooze " + i + " days"; } function setup() { // Create the labels we'll need for snoozing GmailApp.createLabel("Snooze"); for (var i = 1; i <= 7; ++i) { GmailApp.createLabel(getLabelName(i)); } if (ADD_UNSNOOZED_LABEL) { GmailApp.createLabel("Unsnoozed"); } } function moveSnoozes() { var oldLabel, newLabel, page; for (var i = 1; i <= 7; ++i) { newLabel = oldLabel; oldLabel = GmailApp.getUserLabelByName(getLabelName(i)); page = null; // Get threads in "pages" of 100 at a time while(!page || page.length == 100) { page = oldLabel.getThreads(0, 100); if (page.length > 0) { if (newLabel) { // Move the threads into "today's" label newLabel.addToThreads(page); } else { // Unless it's time to unsnooze it GmailApp.moveThreadsToInbox(page); if (MARK_UNREAD) { GmailApp.markThreadsUnread(page); } if (ADD_UNSNOOZED_LABEL) { GmailApp.getUserLabelByName("Unsnoozed") .addToThreads(page); } } // Move the threads out of "yesterday's" label oldLabel.removeFromThreads(page); } } } }
```

Save the script, run “Setup” to create several new labels, and add a trigger to make it run on a daily basis. Messages with “Snooze” label will be marked unread after a certain number of days.

ASSESSMENT QUESTIONS

Q1) WHY DO EMAIL SERVICES READ YOUR EMAIL WHAT IS THEIR GOAL?
Ans) Email services read our emails to look for malware, such as attachments containing viruses, Trojan horses and for preventing phishing attempts. These services then transfer these suspicious emails into a spam folder. Email services state that their main goal is to provide a secure service to the users. However some email services like Gmail also read emails to determine their users' preferences. They then show targeted ads to their users based on their interests that is determined by scanning their emails.

Q2) HOW DOES PGP SECURE EMAIL DIFFERENTLY THEN GMAIL?
Ans) PGP uses a combination of symmetric key encryption and asymmetric key encryption. It does this to combine the speed and efficiency of symmetric encryption with the security of asymmetric key encryption. First PGP generates a single use random session key which can encrypt and decrypt the actual message (*symmetric key encryption*).This random session key is generated by public key given by the recipient to the sender (a*symmetric key encryption*).This session key is then decrypted by the private key unique to the recipient.Now the recipient is able to decrypt the actual message using this session key.
Gmail encrypts the email in transit using a security protocol known as TLS (*transport layer security)*. TLS uses public key infrastructure (PKI) to send the mail securely. When we send an email our browser contacts Google server and establish a secure connection. The message in the email is encrypted and sent to the next trusted server and then decrypted. This process continues until it reaches recipient's server and it thus prevents third party access of the email.
Hence the major difference between PGP and TLS used by Google is that in TLS the email is only secure and encrypted between transmission .However in PGP the data is encrypted and hence secured even when it is not being transmitted. If somehow the public key infrastructure (PKI) is compromised then PGP is more secure since a private key is required to view the actual message.

Q3) WHY PEOPLE DONT USE SERVICES LIKE PGP MORE OFTEN?
Ans) People don't use services like PGP more often because-

- Of the development of end-2-end encryption in instant chat apps
- Due to usability issues meaning lack of balance between security and ease of use. PGP requires setup and technical knowledge thus making it inaccessible to the masses.
- Lack of motivation or knowledge to understand the need of encryption. Since most users don't even have the idea of their data being stolen online and then used against them.
- In some countries it can be considered illegal to encrypt messages hence people usually avoid encryption services.
- Also securing the private key makes it difficult to use it as most of the encryption system depends on it. If the private key is compromised then the whole system fails.

Q4) WHAT IS PHISHING?

Ans) Phishing is a type of social engineering attack in which the attacker pose as a legitimate organization and lures the target/targets to give sensitive information about themselves. This information could include usernames, passwords or credit card details. The attacker can make contact with the victim via phone calls, emails, text messages and social media.

Q5) WHAT IS SPEAR PHISHING?

Ans) Spear phishing is used to target a specific individual or organization by collecting data about the victim. The data is collected by acquiring personal details about the victim online. This data can include friend, employees, emails, location, etc. With this data the attacker can pose as a friend or a familiar entity and send a convincing but fraudulent message to victim. In this way the attacker can obtain sensitive information and use it for malicious reasons .Unlike spear phishing attacks, phishing attacks usually target large and random group of people at the same time.

REFERENCES:

- https://realpython.com
-
- https://gnupg.org/
- https://protonmail.com
- https://www.openpgp.org/
- https://docs.python.org/3/library/email.html
- https://docs.python.org/3/library/smtplib.html
- https://www.wikipedia.org