Name: Pankaj Parihar

Roll No.: 74

Batch: T21

Assignment – 12

Aim : To explore the GPG tool of Linux to implement email security.

Theory :

PGP (Pretty Good Privacy) is an encryption program that provides cryptographic privacy and authentication for data communication. GPG (Gnu Privacy Guard) is a free software replacement for PGP that implements the OpenPGP standard.

PGP using GPG:

1. **Public and Private Key Pair**:
   PGP uses asymmetric encryption, where each user has a **public key** (shared with others) and a **private key** (kept secret). Messages encrypted with the public key can only be decrypted by the private key, and vice versa.

2. **Data Encryption**:

   - **Symmetric Encryption**: For encrypting the actual message, a random symmetric key (session key) is generated. The message is encrypted with this key using a symmetric algorithm like AES.

   - **Asymmetric Encryption**: The session key itself is then encrypted with the recipient's public key using asymmetric encryption (like RSA). This allows the session key to be securely shared.

3. **Digital Signature**:

   - The sender signs the message with their private key, which helps ensure **authenticity** and **integrity**. This digital signature proves the identity of the sender and verifies that the message hasn’t been tampered with.

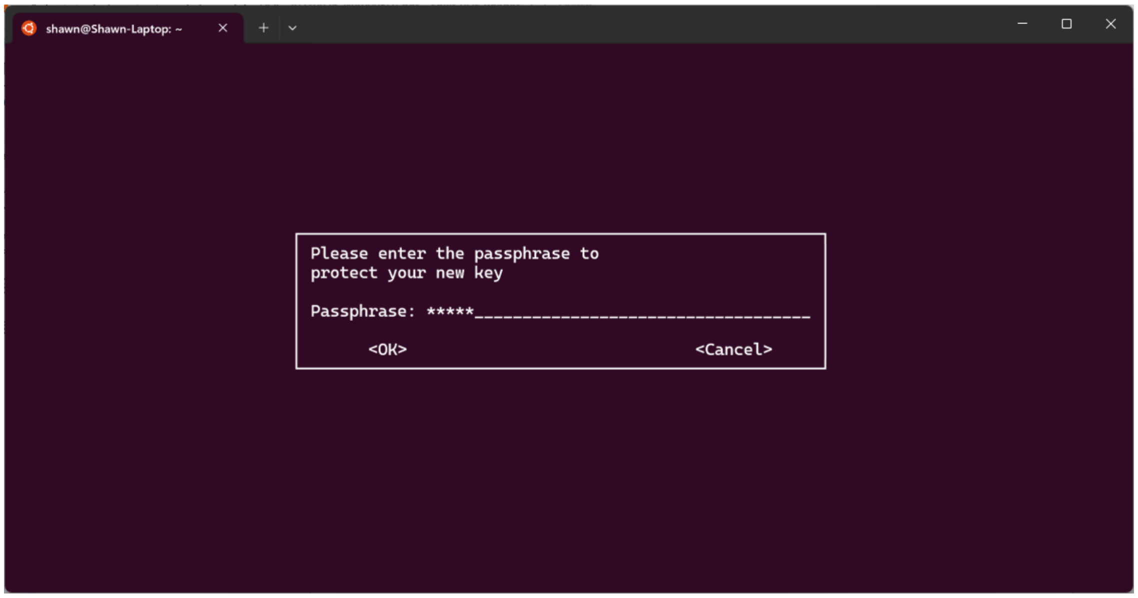
   - The recipient can verify the signature using the sender's public key.

4. **Key Management**:

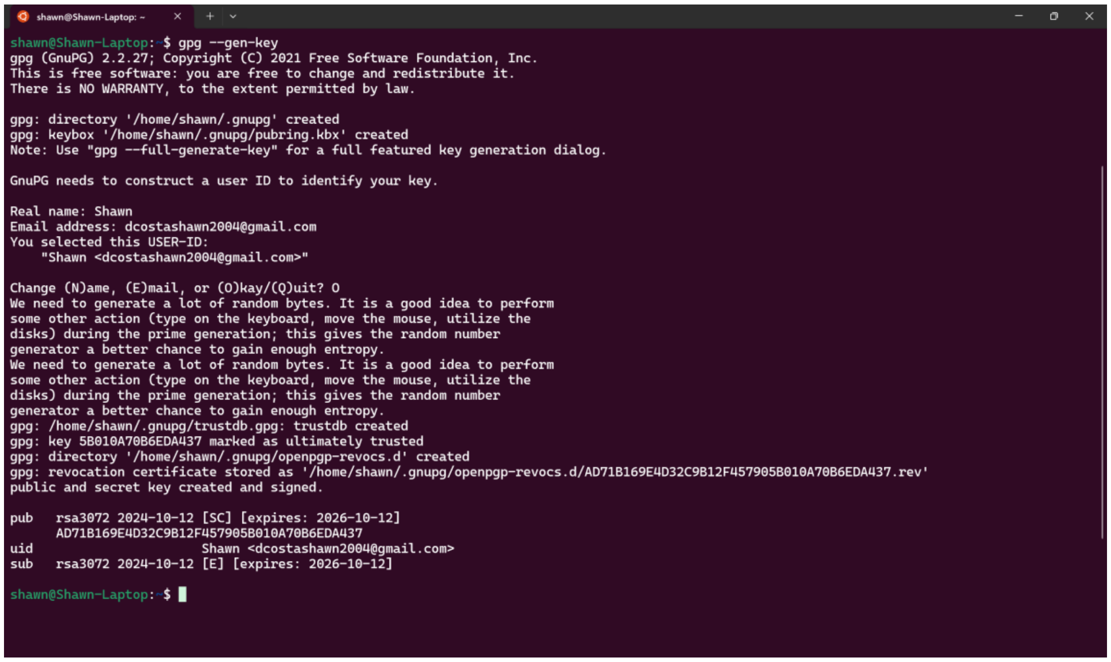
- GPG uses **keyrings** to manage the public and private keys. Users can add trusted public keys and use them to encrypt data.
- GPG also supports **key servers**, where public keys can be shared and retrieved.

5. **Web of Trust**:
    - Unlike centralized systems like SSL, PGP uses a **web of trust** for identity verification. Users sign each other's public keys, building a network of trust without relying on a single central authority.
6. **Message Decryption**:
    - When the recipient receives the message, they use their private key to decrypt the symmetric session key. Once they have the session key, they can decrypt the message.
7. **Passphrase Protection**:
    - To enhance security, private keys are often encrypted with a passphrase, adding an extra layer of protection in case the private key is compromised.

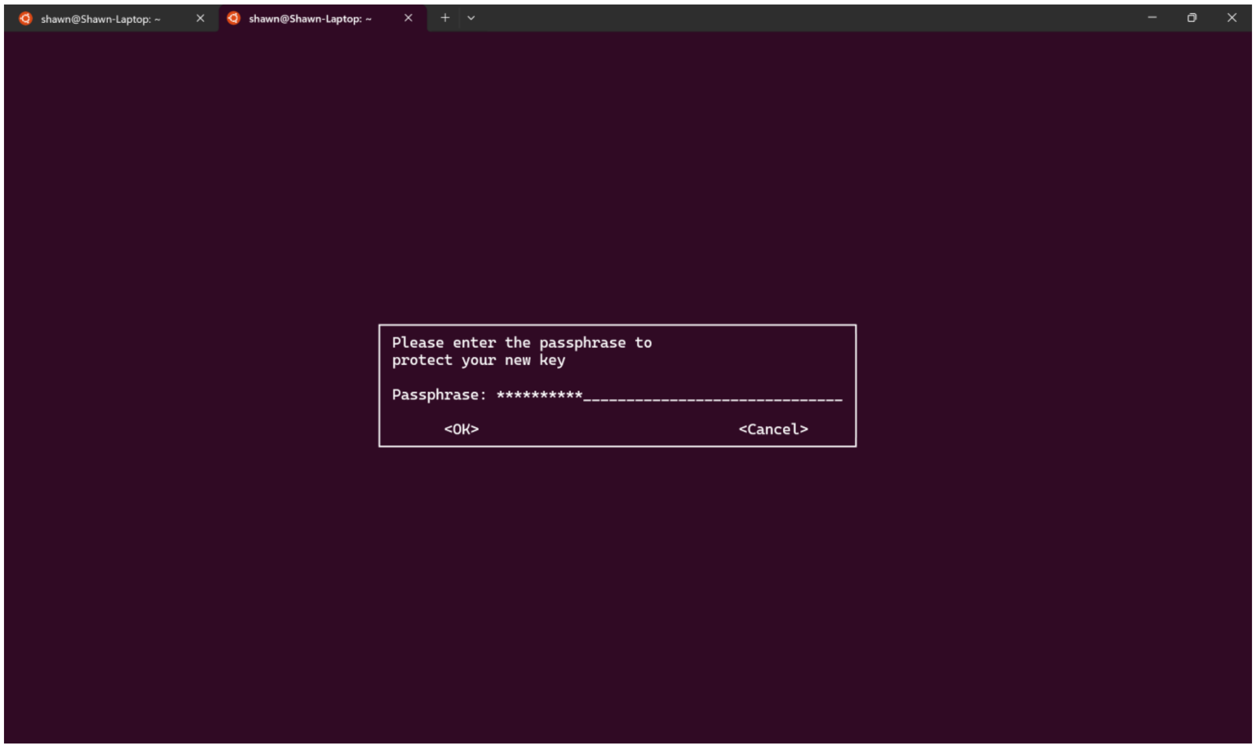
Output:



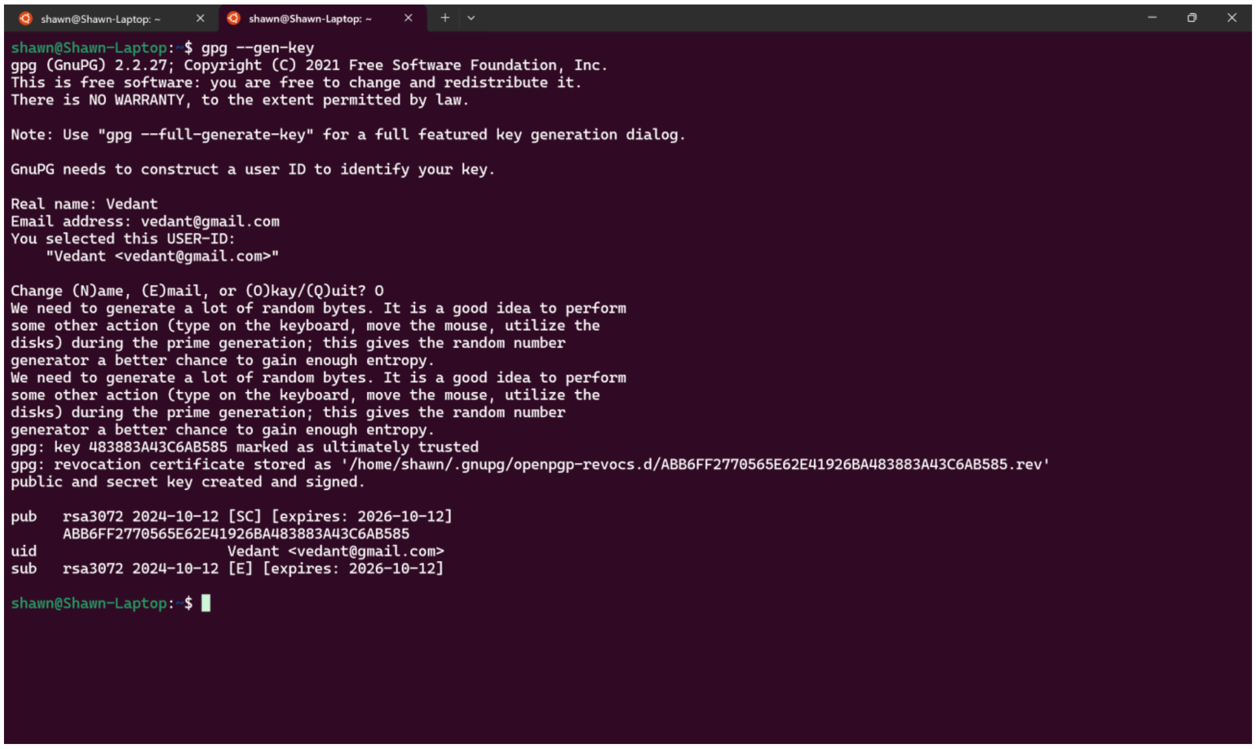
```
Please enter the passphrase to
protect your new key

Passphrase: *****___________________________________

     <OK>                                 <Cancel>
```



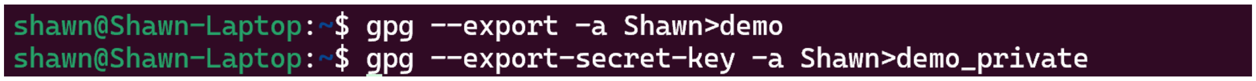
```
shawn@Shawn-Laptop:~$ gpg --gen-key
gpg (GnuPG) 2.2.27; Copyright (C) 2021 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

gpg: directory '/home/shawn/.gnupg' created
gpg: keybox '/home/shawn/.gnupg/pubring.kbx' created
Note: Use "gpg --full-generate-key" for a full featured key generation dialog.

GnuPG needs to construct a user ID to identify your key.

Real name: Shawn
Email address: dcostashawn2004@gmail.com
You selected this USER-ID:
    "Shawn <dcostashawn2004@gmail.com>"

Change (N)ame, (E)mail, or (O)kay/(Q)uit? O
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: /home/shawn/.gnupg/trustdb.gpg: trustdb created
gpg: key 5B010A70B6EDA437 marked as ultimately trusted
gpg: directory '/home/shawn/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/home/shawn/.gnupg/openpgp-revocs.d/AD71B169E4D32C9B12F457905B010A70B6EDA437.rev'
public and secret key created and signed.

pub   rsa3072 2024-10-12 [SC] [expires: 2026-10-12]
      AD71B169E4D32C9B12F457905B010A70B6EDA437
uid                      Shawn <dcostashawn2004@gmail.com>
sub   rsa3072 2024-10-12 [E] [expires: 2026-10-12]

shawn@Shawn-Laptop:~$
```



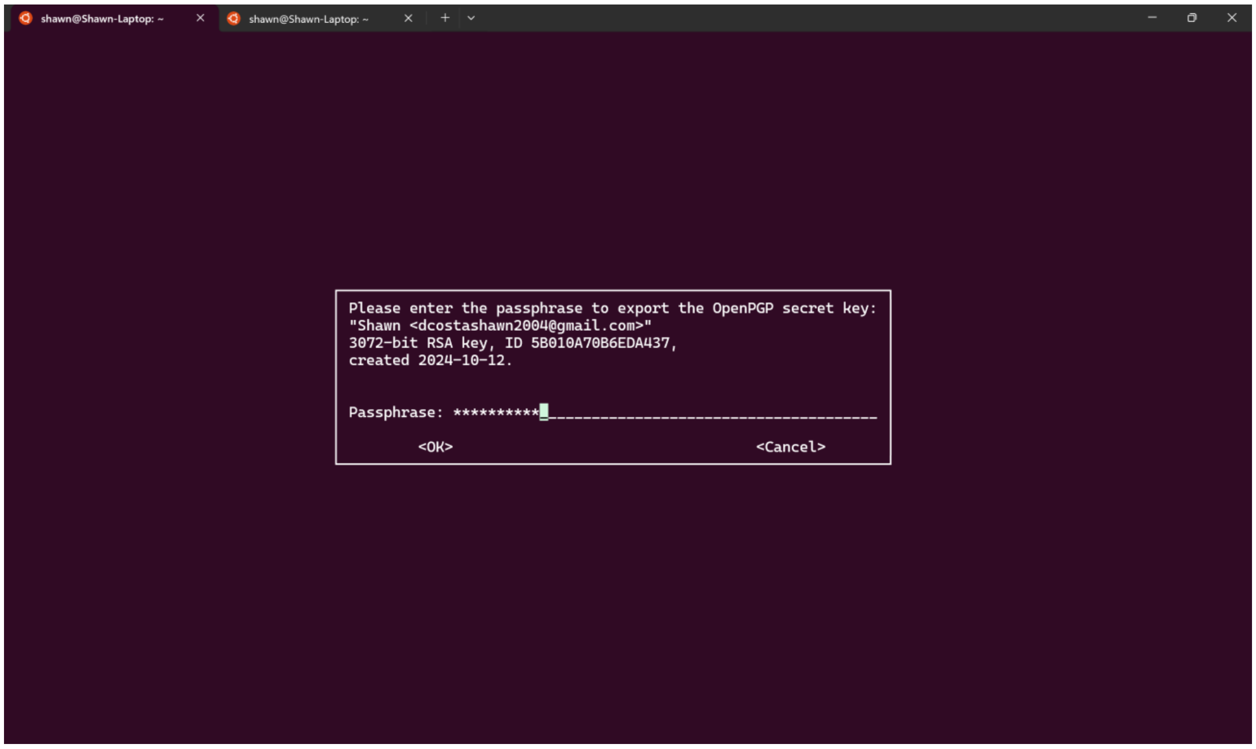
```
Please enter the passphrase to
protect your new key

Passphrase: **********________________________________

     <OK>                               <Cancel>
```



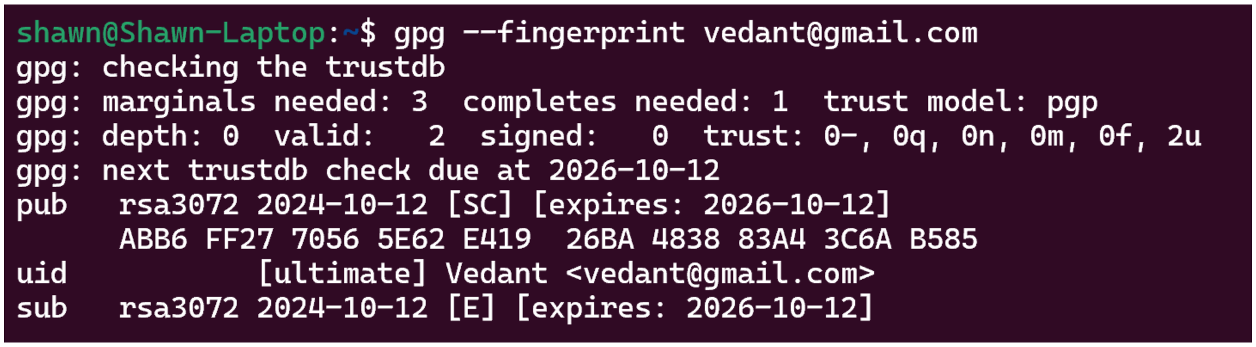
```
shawn@Shawn-Laptop:~$ gpg --gen-key
gpg (GnuPG) 2.2.27; Copyright (C) 2021 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Note: Use "gpg --full-generate-key" for a full featured key generation dialog.

GnuPG needs to construct a user ID to identify your key.

Real name: Vedant
Email address: vedant@gmail.com
You selected this USER-ID:
    "Vedant <vedant@gmail.com>"

Change (N)ame, (E)mail, or (O)kay/(Q)uit? O
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: key 483883A43C6AB585 marked as ultimately trusted
gpg: revocation certificate stored as '/home/shawn/.gnupg/openpgp-revocs.d/ABB6FF2770565E62E41926BA483883A43C6AB585.rev'
public and secret key created and signed.

pub   rsa3072 2024-10-12 [SC] [expires: 2026-10-12]
      ABB6FF2770565E62E41926BA483883A43C6AB585
uid                      Vedant <vedant@gmail.com>
sub   rsa3072 2024-10-12 [E] [expires: 2026-10-12]

shawn@Shawn-Laptop:~$
```



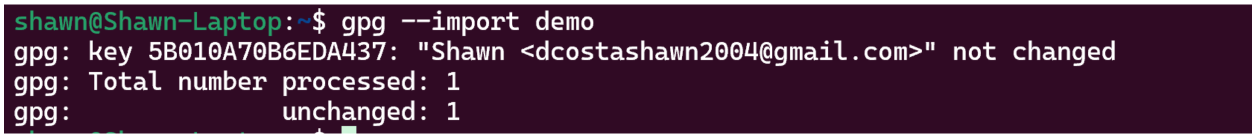
```
shawn@Shawn-Laptop:~$ gpg --export -a Shawn>demo
shawn@Shawn-Laptop:~$ gpg --export-secret-key -a Shawn>demo_private
```



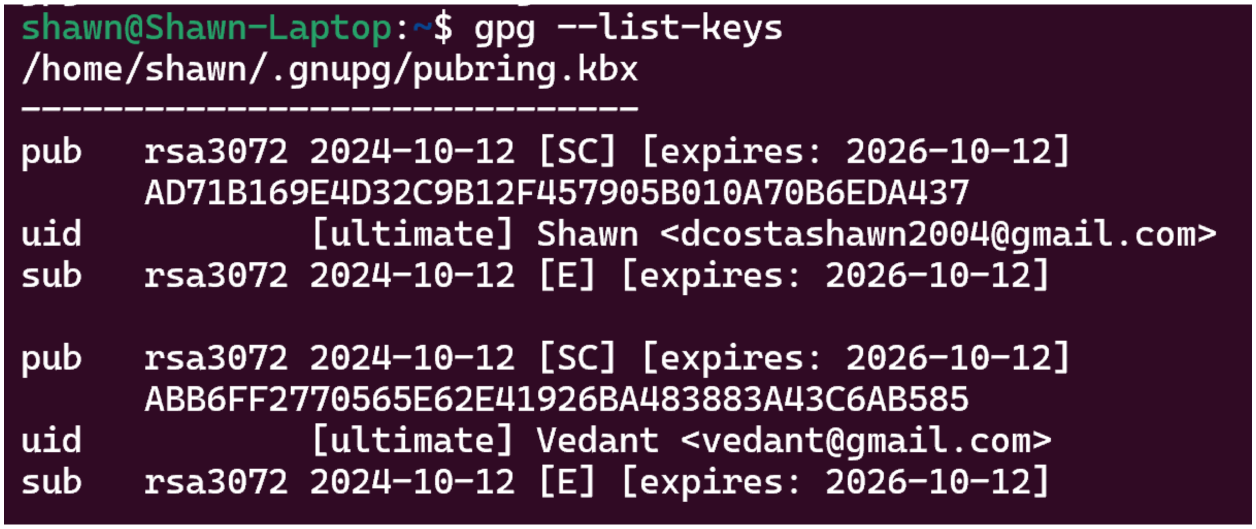
```
Please enter the passphrase to export the OpenPGP secret key:
"Shawn <dcostashawn2004@gmail.com>"
3072-bit RSA key, ID 5B010A70B6EDA437,
created 2024-10-12.

Passphrase: **********_______________________________________

        <OK>                                       <Cancel>
```



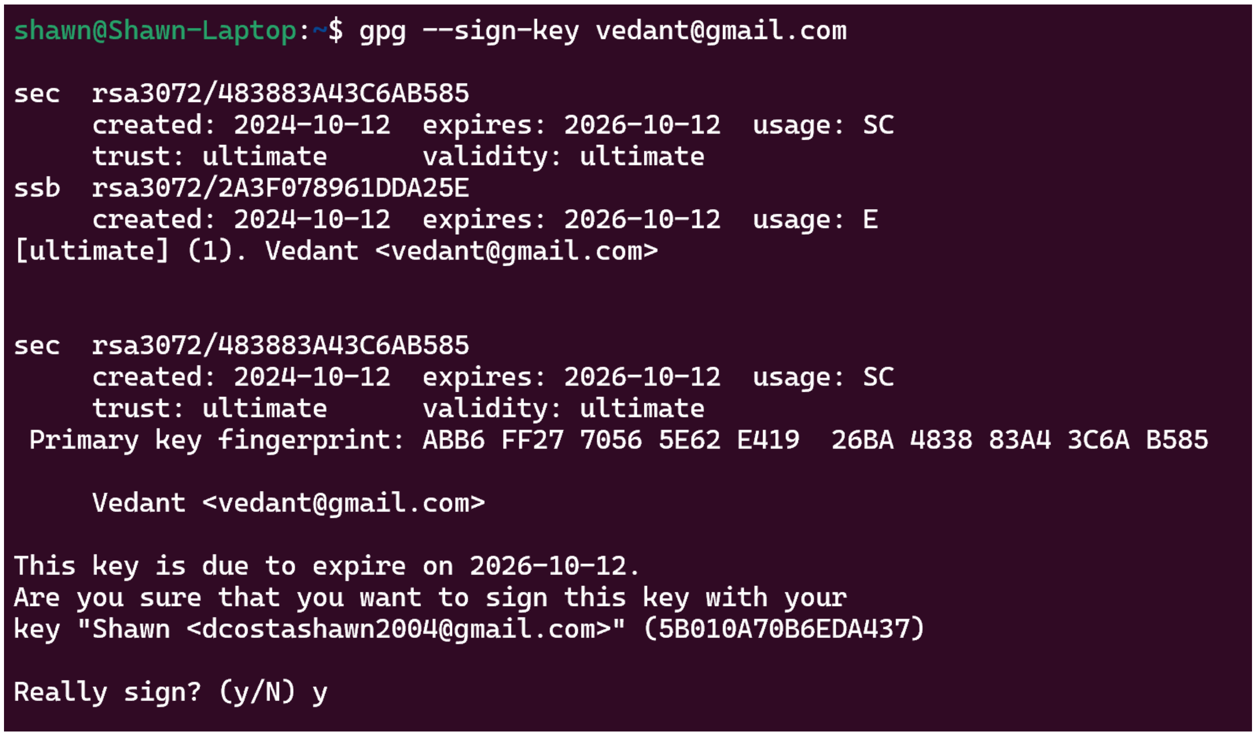
```
shawn@Shawn-Laptop:~$ gpg --fingerprint vedant@gmail.com
gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   2  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 2u
gpg: next trustdb check due at 2026-10-12
pub   rsa3072 2024-10-12 [SC] [expires: 2026-10-12]
      ABB6 FF27 7056 5E62 E419  26BA 4838 83A4 3C6A B585
uid           [ultimate] Vedant <vedant@gmail.com>
sub   rsa3072 2024-10-12 [E] [expires: 2026-10-12]
```



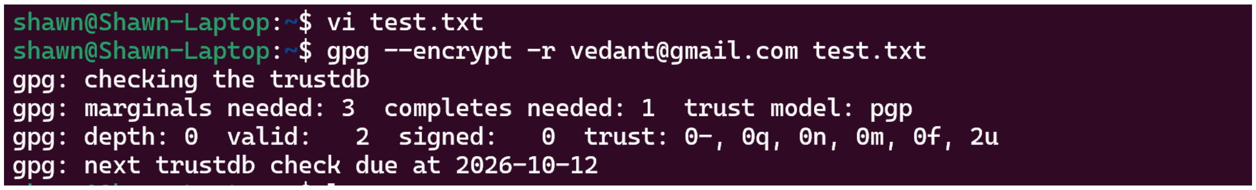
```
shawn@Shawn-Laptop:~$ gpg --import demo
gpg: key 5B010A70B6EDA437: "Shawn <dcostashawn2004@gmail.com>" not changed
gpg: Total number processed: 1
gpg:              unchanged: 1
```



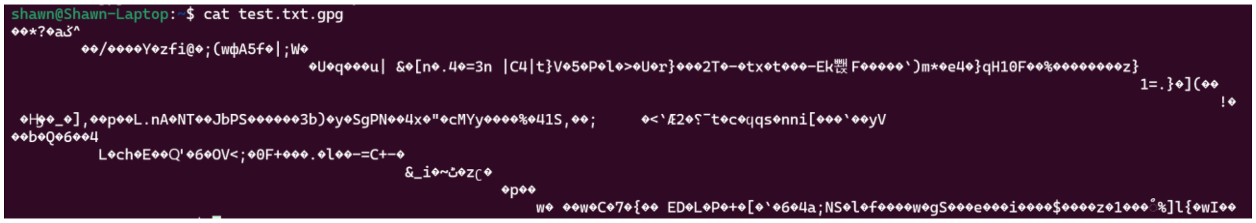
```
shawn@Shawn-Laptop:~$ gpg --list-keys
/home/shawn/.gnupg/pubring.kbx
------------------------------
pub   rsa3072 2024-10-12 [SC] [expires: 2026-10-12]
      AD71B169E4D32C9B12F457905B010A70B6EDA437
uid           [ultimate] Shawn <dcostashawn2004@gmail.com>
sub   rsa3072 2024-10-12 [E] [expires: 2026-10-12]

pub   rsa3072 2024-10-12 [SC] [expires: 2026-10-12]
      ABB6FF2770565E62E41926BA483883A43C6AB585
uid           [ultimate] Vedant <vedant@gmail.com>
sub   rsa3072 2024-10-12 [E] [expires: 2026-10-12]
```



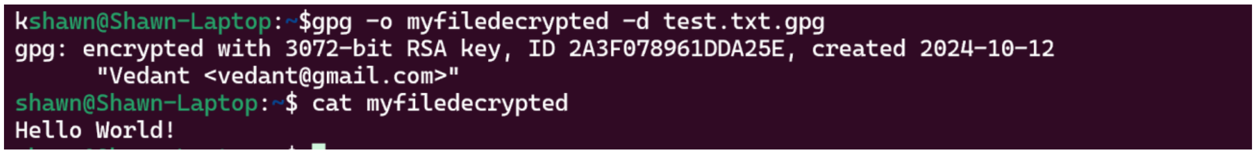
```
shawn@Shawn-Laptop:~$ gpg --sign-key vedant@gmail.com

sec  rsa3072/483883A43C6AB585
     created: 2024-10-12  expires: 2026-10-12  usage: SC
     trust: ultimate      validity: ultimate
ssb  rsa3072/2A3F078961DDA25E
     created: 2024-10-12  expires: 2026-10-12  usage: E
[ultimate] (1). Vedant <vedant@gmail.com>


sec  rsa3072/483883A43C6AB585
     created: 2024-10-12  expires: 2026-10-12  usage: SC
     trust: ultimate      validity: ultimate
 Primary key fingerprint: ABB6 FF27 7056 5E62 E419  26BA 4838 83A4 3C6A B585

     Vedant <vedant@gmail.com>

This key is due to expire on 2026-10-12.
Are you sure that you want to sign this key with your
key "Shawn <dcostashawn2004@gmail.com>" (5B010A70B6EDA437)

Really sign? (y/N) y
```



```
shawn@Shawn-Laptop:~$ vi test.txt
shawn@Shawn-Laptop:~$ gpg --encrypt -r vedant@gmail.com test.txt
gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   2  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 2u
gpg: next trustdb check due at 2026-10-12
```



```
shawn@Shawn-Laptop:~$ cat test.txt.gpg
��*?�aჰ^
        ��/����Y�zfi@�;(wφA5f�|;W�
                                  �U�q���u| &�[n�.4�=3n |C4|t}V�5�P�l�>�U�r}���2T�-�tx�t���-Ek뺇 F�����`)m*�e4�}qH10F��%���������z}
                                                                                                                             1=.}�](��
                                                                                                                                      !�
 �H��_�],��p��L.nA�NT��JbPS������3b)�y�SgPN��4x�"�cMYy����%�41S,��;     �<`Æ2�ʕ¯t�c�qqs�nni[���`��yV
��b�Q�6��4
          L�ch�E��Q'�6�OV<;�0F+���.�l��-=C+-�
                                              &_i�~ථ�z(�
                                                        �p��
                                                            w� ��w�C�7�{�� ED�L�P�+�[�`�6�4a;NS�l�f����w�gS���e���i����$����z�1���˚%]l{�wI��
```



```
kshawn@Shawn-Laptop:~$gpg -o myfiledecrypted -d test.txt.gpg
gpg: encrypted with 3072-bit RSA key, ID 2A3F078961DDA25E, created 2024-10-12
      "Vedant <vedant@gmail.com>"
shawn@Shawn-Laptop:~$ cat myfiledecrypted
Hello World!
```

Conclusion: Demonstrated the network security system using open source tools (LO6 is achieved).