GPG, which stands for GNU Privacy Guard, is a free and open-source encryption software that provides cryptographic privacy and authentication for data communication. It's a modern replacement
for PGP (Pretty Good Privacy) and is widely used for securing emails and files. GPG is available for
various platforms, including Linux, macOS, and Windows. In GPG, the terms "private key ring" and
"public key ring" refer to collections of cryptographic keys used for encryption and decryption:
Private Key Ring:
● The private key ring is a file or storage location where the sender's private keys are stored.
● Private keys are used by the sender for decrypting messages that were encrypted with the receiver's public key and for digitally signing messages or files.
● The sender should guard their private key(s) very carefully because anyone with access to the private key can decrypt their messages and files, impersonate them, and potentially compromise their security.
Public Key Ring:
● The public key ring is a file or storage location where the public keys of the receiver and other potential recipients are stored.
● Public keys are used by the sender for encrypting messages or files that they want to send securely to the receiver. The sender uses the receiver's public key to encrypt the data, and only the receiver can decrypt it with their private key.
● Additionally, public keys are used to verify digital signatures created by the sender or others. If the sender receives a digitally signed message or file, they can use the sender's public key to verify that it was indeed signed by them and hasn't been tampered with.
Commands used for Key Generation and Encryption/Decryption:
Step 1: Generate private key and public key pairs for sender and receiver using
command
gpg --gen-key or gpg –full-generate-key (repeat for sender and receiver)
Step 2: Create a file containing sender’s public key which then can be sent to
other users.
gpg --export -a username>filename (creates file in ascii format) or
gpg --output filename --armor --export user’s_email (for sender)
Step 3: Similarly create a file containing the sender's private key.
gpg --export-secret-key -a username>filename (for sender)
Roll no. : 53
Name: Shreya Kamath
Date: 13th September, 2023.
Step 4: You can create a fingerprint of key using the command
gpg --fingerprint receiver’s_email (for receiver)
Step 5: Sender needs to add in his public key ring, the public key of receiver
(for sender)
gpg --import filename_containing_public_key_of_receiver
Step 6: Listing public keys in keyring
gpg --list-keys (from public key rings of all users)
gpg --list-keys emailid@gmail.com (from public key rings of specific users)
Step 7: Sender can sign the public key of receiver using command
gpg --sign-key receiver_email
Step 8: Encrypt the data to send.
gpg --encrypt -r receiver_email name_of_file
OR
gpg --encrypt --sign --armor -r receiver_email name_of_file
OR

```
gpg --encrypt --sign -r receiver_email name_of_file
```
Step 9: Decrypt the file
```
gpg -o myfiledecrypted -d myfile.txt.gpg
```

10. Explore the GPG tool of linux to implement email security.

```
anchi@lab1003-HP-280-G4-MT-Business-PC:~$ 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: client
Email address: client@gmail.com
You selected this USER-ID:
    "client <client@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 0CCC132BD4DF5838 marked as ultimately trusted
gpg: directory '/home/sanchi/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as
'/home/sanchi/.gnupg/openpgp-revocs.d/C8A38B2C9045A08DDB9003310CCC132BD4DF5838.re
v'
public and secret key created and signed.

pub   rsa3072 2024-10-22 [SC] [expires: 2026-10-22]
      C8A38B2C9045A08DDB9003310CCC132BD4DF5838
uid                      client <client@gmail.com>
sub   rsa3072 2024-10-22 [E] [expires: 2026-10-22]

sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ 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: receiver
Email address: receiver@gmail.com
You selected this USER-ID:
    "receiver <receiver@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 6C30129BFA589429 marked as ultimately trusted
gpg: revocation certificate stored as
'/home/sanchi/.gnupg/openpgp-revocs.d/59DBB104259D320240BF6FB96C30129BFA589429.rev'
public and secret key created and signed.

pub   rsa3072 2024-10-22 [SC] [expires: 2026-10-22]
      59DBB104259D320240BF6FB96C30129BFA589429
uid                      receiver <receiver@gmail.com>
sub   rsa3072 2024-10-22 [E] [expires: 2026-10-22]

sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg --export -a client > client_public_key
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg --export -a receiver > receiver_public_key
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg --export -secert-key -a client >client_private
gpg: conflicting commands
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg --export-secert-key -a client >client_private
invalid option "--export-secert-key"
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg --export -secert-key -a client >client_private
gpg: conflicting commands
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg --export -secert-key -a
receiver>receiver_private
gpg: conflicting commands
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg --export-secre^Ckey -a
receiver>receiver_private
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ ^C
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ ^C
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ 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: sanjana
Email address: sanjana@gmail.com
You selected this USER-ID:
    "sanjana <sanjana@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 5163DE5C283F7D55 marked as ultimately trusted
gpg: revocation certificate stored as
'/home/sanchi/.gnupg/openpgp-revocs.d/A9C7F4FEC2C33A77314DF3865163DE5C283F7D55.rev
'
public and secret key created and signed.

pub   rsa3072 2024-10-22 [SC] [expires: 2026-10-22]
      A9C7F4FEC2C33A77314DF3865163DE5C283F7D55
uid                      sanjana <sanjana@gmail.com>
sub   rsa3072 2024-10-22 [E] [expires: 2026-10-22]

sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ 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: sanchi
Email address: sanchi@gmail.com
You selected this USER-ID:
    "sanchi <sanchi@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 ABBBFE337140F8FB marked as ultimately trusted
gpg: revocation certificate stored as
'/home/sanchi/.gnupg/openpgp-revocs.d/A79159B874617F579D07568CABBBFE337140F8FB.rev'
public and secret key created and signed.

pub   rsa3072 2024-10-22 [SC] [expires: 2026-10-22]
      A79159B874617F579D07568CABBBFE337140F8FB
uid                      sanchi <sanchi@gmail.com>
sub   rsa3072 2024-10-22 [E] [expires: 2026-10-22]

sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg --export -a sanjana>sanjana_public
```

```
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg --export -a sanchi>sanchi_public
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg --export-secret-key -a
sanjana>sanjana_private
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg --export-secret-key -a sanchi>sanchi_private
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg --fingerprint sanchi@gmail.com
gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   4  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 4u
gpg: next trustdb check due at 2026-10-22
pub   rsa3072 2024-10-22 [SC] [expires: 2026-10-22]
      A791 59B8 7461 7F57 9D07  568C ABBB FE33 7140 F8FB
uid           [ultimate] sanchi <sanchi@gmail.com>
sub   rsa3072 2024-10-22 [E] [expires: 2026-10-22]

sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg --import sanchi_public
gpg: key ABBBFE337140F8FB: "sanchi <sanchi@gmail.com>" not changed
gpg: Total number processed: 1
gpg:              unchanged: 1
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg --import sanchi_public
gpg: no valid OpenPGP data found.
gpg: Total number processed: 0
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg --import sanchi_public
gpg: key ABBBFE337140F8FB: "sanchi <sanchi@gmail.com>" not changed
gpg: Total number processed: 1
gpg:              unchanged: 1
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg --import sanchi_public
gpg: key ABBBFE337140F8FB: "sanchi <sanchi@gmail.com>" not changed
gpg: Total number processed: 1
gpg:              unchanged: 1
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg--list-keys
gpg--list-keys: command not found
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg --list-keys
/home/sanchi/.gnupg/pubring.kbx
-------------------------------
pub   rsa3072 2024-10-22 [SC] [expires: 2026-10-22]
      C8A38B2C9045A08DDB9003310CCC132BD4DF5838
uid           [ultimate] client <client@gmail.com>
sub   rsa3072 2024-10-22 [E] [expires: 2026-10-22]

pub   rsa3072 2024-10-22 [SC] [expires: 2026-10-22]
      59DBB104259D320240BF6FB96C30129BFA589429
uid           [ultimate] receiver <receiver@gmail.com>
sub   rsa3072 2024-10-22 [E] [expires: 2026-10-22]

pub   rsa3072 2024-10-22 [SC] [expires: 2026-10-22]
      A9C7F4FEC2C33A77314DF3865163DE5C283F7D55
uid           [ultimate] sanjana <sanjana@gmail.com>
sub   rsa3072 2024-10-22 [E] [expires: 2026-10-22]

pub   rsa3072 2024-10-22 [SC] [expires: 2026-10-22]
      A79159B874617F579D07568CABBBFE337140F8FB
uid           [ultimate] sanchi <sanchi@gmail.com>
```

```
sub   rsa3072 2024-10-22 [E] [expires: 2026-10-22]

sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg --list-keys sanchi@gmail.com
pub   rsa3072 2024-10-22 [SC] [expires: 2026-10-22]
      A79159B874617F579D07568CABBBFE337140F8FB
uid           [ultimate] sanchi <sanchi@gmail.com>
sub   rsa3072 2024-10-22 [E] [expires: 2026-10-22]

sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg --sign-key sanchi@gmail.com

sec  rsa3072/ABBBFE337140F8FB
     created: 2024-10-22  expires: 2026-10-22  usage: SC
     trust: ultimate      validity: ultimate
ssb  rsa3072/A1BBE92EDDA56BDA
     created: 2024-10-22  expires: 2026-10-22  usage: E
[ultimate] (1). sanchi <sanchi@gmail.com>


sec  rsa3072/ABBBFE337140F8FB
     created: 2024-10-22  expires: 2026-10-22  usage: SC
     trust: ultimate      validity: ultimate
 Primary key fingerprint: A791 59B8 7461 7F57 9D07  568C ABBB FE33 7140 F8FB

     sanchi <sanchi@gmail.com>

This key is due to expire on 2026-10-22.
Are you sure that you want to sign this key with your
key "client <client@gmail.com>" (0CCC132BD4DF5838)

Really sign? (y/N) n

Key not changed so no update needed.
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg --delete-secret-key ^C
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg --delete-secret-key
OCCC132BD4D4DF5838
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: key "OCCC132BD4D4DF5838" not found: Not found
gpg: OCCC132BD4D4DF5838: delete key failed: Not found
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg --delete-secret-key A1BBE92EDDA56BDA
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.


sec  rsa3072/ABBBFE337140F8FB 2024-10-22 sanchi <sanchi@gmail.com>

Delete this key from the keyring? (y/N) n
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg --delete-secret-key ABBBFE337140F8FB
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.


sec  rsa3072/ABBBFE337140F8FB 2024-10-22 sanchi <sanchi@gmail.com>

Delete this key from the keyring? (y/N) n
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg --list-secret-keys
/home/sanchi/.gnupg/pubring.kbx
------------------------------
sec   rsa3072 2024-10-22 [SC] [expires: 2026-10-22]
      C8A38B2C9045A08DDB9003310CCC132BD4DF5838
uid           [ultimate] client <client@gmail.com>
ssb   rsa3072 2024-10-22 [E] [expires: 2026-10-22]

sec   rsa3072 2024-10-22 [SC] [expires: 2026-10-22]
      59DBB104259D320240BF6FB96C30129BFA589429
uid           [ultimate] receiver <receiver@gmail.com>
ssb   rsa3072 2024-10-22 [E] [expires: 2026-10-22]

sec   rsa3072 2024-10-22 [SC] [expires: 2026-10-22]
      A9C7F4FEC2C33A77314DF3865163DE5C283F7D55
uid           [ultimate] sanjana <sanjana@gmail.com>
ssb   rsa3072 2024-10-22 [E] [expires: 2026-10-22]

sec   rsa3072 2024-10-22 [SC] [expires: 2026-10-22]
      A79159B874617F579D07568CABBBFE337140F8FB
uid           [ultimate] sanchi <sanchi@gmail.com>
ssb   rsa3072 2024-10-22 [E] [expires: 2026-10-22]

sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg --delete-secret-key
C8A38B2C9045A08DDB9003310CCC132BD4DF5838
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.


sec  rsa3072/0CCC132BD4DF5838 2024-10-22 client <client@gmail.com>

Delete this key from the keyring? (y/N) y
This is a secret key! - really delete? (y/N) y
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg --sign-key sanchi@gmail.com

sec  rsa3072/ABBBFE337140F8FB
     created: 2024-10-22  expires: 2026-10-22  usage: SC
     trust: ultimate      validity: ultimate
ssb  rsa3072/A1BBE92EDDA56BDA
     created: 2024-10-22  expires: 2026-10-22  usage: E
[ultimate] (1). sanchi <sanchi@gmail.com>


sec  rsa3072/ABBBFE337140F8FB
```

```
     created: 2024-10-22  expires: 2026-10-22  usage: SC
     trust: ultimate      validity: ultimate
 Primary key fingerprint: A791 59B8 7461 7F57 9D07  568C ABBB FE33 7140 F8FB

     sanchi <sanchi@gmail.com>

This key is due to expire on 2026-10-22.
Are you sure that you want to sign this key with your
key "receiver <receiver@gmail.com>" (6C30129BFA589429)

Really sign? (y/N) n

Key not changed so no update needed.
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg --delete-secret-key
59DBB104259D320240BF6FB96C30129BFA589429
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.


sec  rsa3072/6C30129BFA589429 2024-10-22 receiver <receiver@gmail.com>

Delete this key from the keyring? (y/N) y
This is a secret key! - really delete? (y/N) y
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg --sign-key sanchi@gmail.com

sec  rsa3072/ABBBFE337140F8FB
     created: 2024-10-22  expires: 2026-10-22  usage: SC
     trust: ultimate      validity: ultimate
ssb  rsa3072/A1BBE92EDDA56BDA
     created: 2024-10-22  expires: 2026-10-22  usage: E
[ultimate] (1). sanchi <sanchi@gmail.com>


sec  rsa3072/ABBBFE337140F8FB
     created: 2024-10-22  expires: 2026-10-22  usage: SC
     trust: ultimate      validity: ultimate
 Primary key fingerprint: A791 59B8 7461 7F57 9D07  568C ABBB FE33 7140 F8FB

     sanchi <sanchi@gmail.com>

This key is due to expire on 2026-10-22.
Are you sure that you want to sign this key with your
key "sanjana <sanjana@gmail.com>" (5163DE5C283F7D55)

Really sign? (y/N) y

sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg--encrypt -r sanchi@gmail.com sanchi.gpg
gpg--encrypt: command not found
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg --encrypt -r sanchi@gmail.com sanchi.gpg
gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
```

```
gpg: depth: 0  valid:   4  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 4u
gpg: next trustdb check due at 2026-10-22
gpg: can't open 'sanchi.gpg': No such file or directory
gpg: sanchi.gpg: encryption failed: No such file or directory
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg --encrypt --sign -armor -r
sanchi@gmail.com sanchi.gpg
gpg: mor: skipped: No public key
gpg: sanchi.gpg: sign+encrypt failed: No public key
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg --encrypt -r sanchi@gmail.com sanchi
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg --encrypt --sign -armor -r
sanchi@gmail.com sanchi.gpg
gpg: mor: skipped: No public key
gpg: sanchi.gpg: sign+encrypt failed: No public key
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg --encrypt--sign -armor -r sanchi@gmail.com
sanchi.gpg
invalid option "--encrypt--sign"
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg --encrypt --sign --armor -r
sanchi@gmail.com sanchi.gpg
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg -0 sanchidecrypted -d sanchi.txt.gpg
invalid option "-0"
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg -o sanchidecrypted -d sanchi.txt.gpg
gpg: can't open 'sanchi.txt.gpg': No such file or directory
gpg: decrypt_message failed: No such file or directory
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$ gpg -o sanchidecrypted -d sanchi.gpg
gpg: encrypted with 3072-bit RSA key, ID A1BBE92EDDA56BDA, created 2024-10-22
      "sanchi <sanchi@gmail.com>"
sanchi@lab1003-HP-280-G4-MT-Business-PC:~$
```