

ASSIGNMENT 12

AIM: Explore the GPG tool of linux to implement email security.

LO MAPPED: LO6

THEORY:

GPG is the OpenPGP part of the GNU Privacy Guard (GnuPG). It is a tool to provide digital encryption and signing services using the OpenPGP standard. gpg features complete key management and all the bells and whistles you would expect from a full OpenPGP implementation.

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)**

```

Lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ gpg --gen-key
gpg (GnuPG) 2.2.4; Copyright (C) 2017 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: DeepNoob
Email address: deep2003prajapati@gmail.com
You selected this USER-ID:
  "DeepNoob <deep2003prajapati@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 8C5F81EF6E105DAD marked as ultimately trusted
gpg: revocation certificate stored as '/home/lab1006/.gnupg/openpgp-revocs.d/E2500E
1E09CAB680CC5C3B278C5F81EF6E105DAD.rev'
public and secret key created and signed.

pub   rsa3072 2023-10-12 [SC] [expires: 2025-10-11]
       E2500E1E09CAB680CC5C3B278C5F81EF6E105DAD
uid     DeepNoob <deep2003prajapati@gmail.com>
sub     rsa3072 2023-10-12 [E] [expires: 2025-10-11]

Lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ gpg --gen-key
gpg (GnuPG) 2.2.4; Copyright (C) 2017 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: anarky
Email address: akshay3002rathod@gmail.com
You selected this USER-ID:
  "anarky <akshay3002rathod@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 84DCD9EA68B82D0C marked as ultimately trusted
gpg: revocation certificate stored as '/home/lab1006/.gnupg/openpgp-revocs.d/E9A419
D5E025EA52C38940984DCD9EA68B82D0C.rev'
public and secret key created and signed.

pub   rsa3072 2023-10-12 [SC] [expires: 2025-10-11]
       E9A419D5E025EA52C38940984DCD9EA68B82D0C
uid     anarky <akshay3002rathod@gmail.com>
sub     rsa3072 2023-10-12 [E] [expires: 2025-10-11]

```

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)**

```

Lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ gpg --export -a anarky>anarky_public

```

Step 3: Similarly create file containing sender's private key. `gpg --export-`

`secret-key -a username>filename` **(for sender)**

```

Lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ gpg --export-secret-key -a anarky>anark_y_pvt

```

Step 4: You can create a fingerprint of key using the command `gpg --fingerprint receiver's_email` (for receiver)

```
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ gpg --export -a anarky>anarky_public
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ gpg --export -a deepnoob>deepnoob_public
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ gpg --fingerprint deep2003prajapati@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 2025-10-05
pub   rsa3072 2023-10-12 [SC] [expires: 2025-10-11]
       E250 0E1E 09CA 8680 CC5C 3B27 8C5F 81EF 6E10 5DAD
uid    [ultimate] DeepNoob <deep2003prajapati@gmail.com>
pub   rsa3072 2023-10-12 [SC] [expires: 2025-10-11]
```

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`

```
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ gpg --import deepnoob_public
gpg: key 8C5F81EF6E105DAD: "DeepNoob <deep2003prajapati@gmail.com>" not changed
gpg: Total number processed: 1
gpg:      unchanged: 1
```

Step 6: Listing public keys in keyring `gpg --list-key` (from public key rings of all users) `gpg --list-keys shachi_natu@yahoo.com` (from public key rings of specific users)

```
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ gpg --list-key
/home/lab1006/.gnupg/pubring.kbx
-----
pub   rsa3072 2023-10-06 [SC] [expires: 2025-10-07]
```

Step 7: Sender can sign the public key of receiver using command
`gpg --sign-key receiver_email`

```
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ gpg --sign-key deep2003prajapati@gmail.com
sec  rsa3072/8C5F81EF6E105DAD
      created: 2023-10-12  expires: 2025-10-11  usage: SC
      trust: ultimate    validity: ultimate
ssb  rsa3072/24D67A3CC36F9241
      created: 2023-10-12  expires: 2025-10-11  usage: E
[ultimate] (1), DeepNoob <deep2003prajapati@gmail.com>

sec  rsa3072/8C5F81EF6E105DAD
      created: 2023-10-12  expires: 2025-10-11  usage: SC
      trust: ultimate    validity: ultimate
Primary key fingerprint: E250 0E1E 09CA 8680 CC5C 3B27 8C5F 81EF 6E10 5DAD
DeepNoob <deep2003prajapati@gmail.com>

This key is due to expire on 2025-10-11.
Are you sure that you want to sign this key with your
key "anarky <akashay3002rathod@gmail.com>" (84DCD9EA6B082D0C)
Really sign? (y/N) y
```

When you sign the key, it means you verify that you trust the person is who they claim to be. This can help other people decide whether to trust that person too. If someone trusts you, and they see that you've signed this person's key, they may be more likely to trust their identity too.

Step 8: Encrypt the data to send. (create a file beforehand to be encrypted) `gpg --encrypt -r receiver_email name_of_file` **(only encrypt, .gpg file created)**

OR

`gpg --encrypt --sign --armor -r receiver_email name_of_file`

(encrypt and sign, ascii file created)

OR

`gpg --encrypt --sign -r receiver_email name_of_file`

(encrypt and sign, .gpg file created)

```
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ gpg --encrypt -r deep2003prajapati@gmail.com anarky.gpg
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 2025-10-11
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$
```

Step 9: Decrypt the file

`gpg -o myfiledecrypted -d myfile.txt.gpg`

```
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ gpg -o myfiledecrypted -d anarky.gpg
gpg: no valid OpenPGP data found.
gpg: decrypt_message failed: Unknown system error
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ gpg -o anarky -d anarky.gpg
gpg: no valid OpenPGP data found.
gpg: decrypt_message failed: Unknown system error
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ gpg -o myfiledecrypted -d anarky.gpg.gpg
gpg: encrypted with 3072-bit RSA key, ID 24D67A3CC36F9241, created 2023-10-12
"DeepNoob <deep2003prajapati@gmail.com>"
File 'myfiledecrypted' exists. Overwrite? (y/N) n
Enter new filename: xsdjfexdtmy,l.';
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$ gpg -d -o anarky.gpg anarky.gpg.gpg
gpg: encrypted with 3072-bit RSA key, ID 24D67A3CC36F9241, created 2023-10-12
"DeepNoob <deep2003prajapati@gmail.com>"
File 'anarky.gpg' exists. Overwrite? (y/N) y
lab1006@lab1006-HP-280-G4-MT-Business-PC:~$
```

CONCLUSION: In this assignment we have explored the GPG tool of linux and implemented email security.