**AIM: To study and explore the GPG tool of linux to implement email security.**

**THEORY:**

**Hardware / Software Required**: Unix/Linux, gnupg

**What is PGP and GnuPG?**

Pretty Good Privacy (PGP) is a data encryption and decryption computer program that provides cryptographic privacy and authentication for data communication. PGP is often used for signing, encrypting, and decrypting texts, e-mails, files, directories, and whole disk partitions and to increase the security of e-mail communications. PGP encryption uses a serial combination of hashing, data compression, symmetric-key cryptography, and finally public-key cryptography; each step uses one of several supported algorithms. Each public key is bound to a user name and/or an e-mail address. The first version of this system was generally known as a web of trust to contrast with the X.509 system, which uses a hierarchical approach based on certificate authority and which was added to PGP implementations later. Current versions of PGP encryption include both options through an automated key management server. GNU Privacy Guard (GnuPG or GPG) is a free software replacement for Symantec's PGP cryptographic software suite. GnuPG is a hybrid-encryption software program because it uses a combination of conventional symmetric-key cryptography for speed, and public-key cryptography for ease of secure key exchange, typically by using the recipient's public key to encrypt a session key which is only used once. This mode of operation is part of the OpenPGP standard and has been part of PGP from its first version.

**CODE:**

**pc@pc-Lenovo-S510:~$ sudo apt-get install gnupg**

[sudo] password for pc:

Reading package lists... Done

Building dependency tree

Reading state information... Done

gnupg is already the newest version (2.2.4-1ubuntu1.2).

0 upgraded, 0 newly installed, 0 to remove and 47 not upgraded.

**pc@pc-Lenovo-S510:~$ 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: Harry Potter

Email address: HarryPotter@gmail.com

You selected this USER-ID:

"Harry Potter <HarryPotter@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 39E17DFAB1F38A1D marked as ultimately trusted

gpg: directory '/home/pc/.gnupg/openpgp-revocs.d' created

gpg: revocation certificate stored as '/home/pc/.gnupg/openpgp-revocs.d/15891E546DB99C33726FAEA139E17DFAB1F38A1D.rev'

public and secret key created and signed.

pub rsa3072 2020-03-12 [SC] [expires: 2022-03-12]

15891E546DB99C33726FAEA139E17DFAB1F38A1D

uid Harry Potter <HarryPotter@gmail.com>

sub rsa3072 2020-03-12 [E] [expires: 2022-03-12]

**pc@pc-Lenovo-S510:~$ gpg --list-keys**

gpg: checking the trustdb

gpg: marginals needed: 3 completes needed: 1 trust model: pgp

gpg: depth: 0 valid: 1 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, 1u

gpg: next trustdb check due at 2022-03-12

/home/pc/.gnupg/pubring.kbx

---------------------------

pub rsa3072 2020-03-12 [SC] [expires: 2022-03-12]

15891E546DB99C33726FAEA139E17DFAB1F38A1D

uid [ultimate] Harry Potter <HarryPotter@gmail.com>

sub rsa3072 2020-03-12 [E] [expires: 2022-03-12]

**pc@pc-Lenovo-S510:~$ gpg --armor --export HarryPotter@gmail.com>mypk**

**pc@pc-Lenovo-S510:~$ cat mypk**

-----BEGIN PGP PUBLIC KEY BLOCK-----

mQGNBF5p4YQBDAC1sAX2HCD2tytIRV8GZEqTc8V23A0GD0iI5ntSk/s3yP6rMoex

KIzqk2+HlDotHWckHpi1Q7lfE+8TVuuxd8cP7Csst45xvsPy/oxmFhBr+6PDZhGQ

SlOoSY39VhvqEIpzS/jdjad8vf5T4stg9oElNTngz82KHq4f7Kb9tTHeG1pgYIdE

Gg5f/ytfm4TEbWTuZSHQ7GCP+6yaQZJR9A/sPEHGigMIGwuZMNEOjGPX+XZ6Q8Nt

93eaiJog00Td+OxrdfILeAFD1eFrt448dnC+5x1Hai5qx7jrbpTwCPDdTSPa5XVK

RIW9J4opJ6YBoSPWDA6BpcXlakNWRvX3/WyBPFu1SW3iqCwf0CKfL3MqFOtKoUCO

fH6/FV+IudjGXppJJdnbcsc4utZalUZi7Zpvk87Wv/ljlwqGyITdlitbsXa8ij/a

FG4qR8dRzmjTZVETc4MFI90KA9TktmN731g8vA8KQvkPxdAGBdWrgr0AXhKSyCV4

2N/c1EoKQVcIUrcAEQEAAbQkSGFycnkgUG90dGVyIDxoYXJyeXBvdHRlckBnbWFp

bC5jb20+iQHUBBMBCgA+FiEEFYkeVG25nDNyb66hOeF9+rHzih0FAl5p4YQCGwMF

CQPCZwAFCwkIBwIGFQoJCAsCBBYCAwECHgECF4AACgkQOeF9+rHzih2x1Av9GBvc

Iw2parQqLzMMLfktD5m4OG1w1HDrzvTyk8XC4+e1QkBQoYw4niw/sxsMBrYPVzxR

y7iD265ysarnUBkdUmG692/bb1s6S2gLKSEwtjFxXAWuVYAxeWZ1+oQlc0ytab+v

+tWBZSV7PgRq5fwfjMl830RMJqoefmYSaEwyDZS4RrWXtja/OiVU/vdzRHZjYD18

MAZTzP5cJ0Ah4dc1WhYvCj8iaAXcgkJ4MZUVNIOX+/W2OrC0XZIYKU5ps86gf5j6

XTBm/0SUEjdshpX9FeQdtAag4b53L4Y4UUoeucczOJRHd0S3bYmv2OyvSmvAOlAY

mg90PbVx5oPadUX+c7QCBpQWbx8WoFmoYFk3JHsH7CHyA2nlrIpSg7aAboF3uGL8

S06LdY6ZzaPxQNQDdzHDcUo2NYe53MJZuqspuYUAd3jfLMrtaEUhxHc6Eje5Fzto

SRGsKHBHquCUi8+nrZ3FWQnexrS5HV35/sHrrW7iA5Xw76WoRFdUAlJjLCtCuQGN

BF5p4YQBDADW7De/2xhFEdNfnjw2fbJvECTKqbBdg3Brc+TMWYRF5Fzb4asxK1RL

tPzSbaG/KdQ6Q5jFc0RWPsOtXifqxkcy3A0OdSKkzSuUBHkItxw5oJjy/V6wTc84

1ZGKZC6aso1q5GR3galmRb+sxZumHLzbuXpMPLH3zga7QS4Ajlp6wBJd8r56e29q

6n9/nx77gpifiouc8STpzlyN96+Cj+YLYaawak7Q6F1p2BSb1HVGbRR1H2G9KrTz

r30frA5F5Icisnq7xtt8uo8e/MV3EvLYyGXvhb2FW2GA4Hd8T5WjM8gyyRtZjGyw

Mog1mqI2wn4gv5axoAOJxnXekfjwMHP8JrVS0g+cr1Clcn+wcvKiPQ7Y7C+s6X9j

M1OIeK7un0P8W+2nIFlMblIoYY0sfrj0XTMRv3R0MX+dZhf02AQaJdx6PJhAWZlb

XHjhOhKd1lV3vUGFoMFU0uVXcOjQotymQlid6X0lfirkoJQB8jXK830zUwsxtVad

7z4sCZGc+zcAEQEAAYkBvAQYAQoAJhYhBBWJHlRtuZwzcm+uoTnhffqx84odBQJe

aeGEAhsMBQkDwmcAAAoJEDnhffqx84odQqcL/jR5Ll/c5FjY1mggSvFMbyXj1ZNN

eXFOvEOo8BlcM57Epi20MVIFL3RoA9oLTwKVVvPOyCReFyolFj3QeS7Gcb8TzTDv

8E3h1DIigV59gzTD7jg1pPRqi3a5kGJ+/eXbBLrck79/VgoCuYkGvS2OiuL63Eya

9fNpi7Ey/zmmvqN4PBwXntPrGB+bWixBkzKtqvZ1lMUoFPLzDEysZQdnKGIriOck

ifLxV1LepKpj+ikd3TBjBOTr5BIi71VtC0dolB4qJAlH/0Kp1bIaW2xqCfU3y7Jo

xdv2cWpM5wvZY9Zu2vI+BhAXacg8sGBMdSwOcDmdh4Kf2Qe4xJynlOEmKawoeCUv

3tuF0v///PfBgb3PuDazIT28xHrGZILtjgoyob5MnVkCKgQPHtxd9/AnU6R2sxYB

IH9YvIJVzfLaUAhf6vLGtE4V8a99Nbh8Ecx2P50J6it/aPnlEPN85WVkUu+xeHHD

P8XGO5X8z5ke7oexCu3SqF5N6KrC2iyVp5Uo6g==

=M/vK

-----END PGP PUBLIC KEY BLOCK-----

**pc@pc-Lenovo-S510:~$ 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: Ken Adams

Email address: kenadams@gmail.com

You selected this USER-ID:

"Ken Adams <kenadams@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 3DDAFD735B07F8C1 marked as ultimately trusted

gpg: revocation certificate stored as '/home/pc/.gnupg/openpgp-revocs.d/F3273FF326E210265697993B3DDAFD735B07F8C1.rev'

public and secret key created and signed.

pub rsa3072 2020-03-12 [SC] [expires: 2022-03-12]

F3273FF326E210265697993B3DDAFD735B07F8C1

uid Ken Adams <kenadams@gmail.com>

sub rsa3072 2020-03-12 [E] [expires: 2022-03-12]

**pc@pc-Lenovo-S510:~$ gpg --list-keys**

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 2022-03-12

/home/pc/.gnupg/pubring.kbx

---------------------------

pub rsa3072 2020-03-12 [SC] [expires: 2022-03-12]

15891E546DB99C33726FAEA139E17DFAB1F38A1D

uid [ultimate] Harry Potter <HarryPotter@gmail.com>

sub rsa3072 2020-03-12 [E] [expires: 2022-03-12]

pub rsa3072 2020-03-12 [SC] [expires: 2022-03-12]

F3273FF326E210265697993B3DDAFD735B07F8C1

uid [ultimate] Ken Adams <kenadams@gmail.com>

sub rsa3072 2020-03-12 [E] [expires: 2022-03-12]

**pc@pc-Lenovo-S510:~$ gpg --import mypk**

gpg: key 39E17DFAB1F38A1D: "Harry Potter <HarryPotter@gmail.com>" not changed

gpg: Total number processed: 1

gpg: unchanged: 1

**pc@pc-Lenovo-S510:~$ gpg --edit-key HarryPotter@gmail.com**

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.

Secret key is available.

sec rsa3072/39E17DFAB1F38A1D

created: 2020-03-12 expires: 2022-03-12 usage: SC

trust: ultimate validity: ultimate

ssb rsa3072/DCEB9A0F63CA5C1B

created: 2020-03-12 expires: 2022-03-12 usage: E

[ultimate] (1). Harry Potter <HarryPotter@gmail.com>

**gpg> fpr**

pub rsa3072/39E17DFAB1F38A1D 2020-03-12 Harry Potter <HarryPotter@gmail.com>

Primary key fingerprint: 1589 1E54 6DB9 9C33 726F AEA1 39E1 7DFA B1F3 8A1D

**gpg> sign**

"Harry Potter <HarryPotter@gmail.com>" was already signed by key 39E17DFAB1F38A1D

Nothing to sign with key 39E17DFAB1F38A1D

**gpg> quit**

**pc@pc-Lenovo-S510:~$ cat > secrets**

hello

how r u?^C

**pc@pc-Lenovo-S510:~$ cat secrets**

hello

how r u?

**pc@pc-Lenovo-S510:~$ gpg --out secrets\_san --encrypt secrets**

You did not specify a user ID. (you may use "-r")

Current recipients:

Enter the user ID. End with an empty line: HarryPotter@gmail.com

Current recipients:

rsa3072/DCEB9A0F63CA5C1B 2020-03-12 "Harry Potter <[HarryPotter@gmail.com](mailto:harrypotter@gmail.com)>"

**pc@pc-Lenovo-S510:~$ ls**

add.tcl Documents examples.desktop mypk Pictures secrets simple.tcl Templates Videos

Desktop Downloads Music secrets\_san out.nam Public semil.py snap test.c

**pc@pc-Lenovo-S510:~$ cat secrets\_san**

�# �k�F��ӥ#�D=7�P��#�%#0@��#��C�-bQYz� ؆�|�#!�E�h�� k6\#h��#g�$�������i���YBSxeZ���� v�#M�m5P�D�"�U�w�x�����i�Sѣ�� ��>�#��#}t �3�e#���ނ�#r��A�W.�o#t "�wU#YZ#��V������9��#(m�&v�l&��`�#>)���© �1(&6���H�E���Fa�`ls&�{m4E���Q#� mn��1�� ��i�,�fO$�;�d#��S# **pc@pc-Lenovo-S510:~$**

**pc@pc-Lenovo-S510:~$ gpg --output secrets\_from\_san --decrypt secrets\_san**

You need a passphrase to unlock the secret key for

user: "Harry Potter <HarryPotter@gmail.com>"

4096-bit RSA key, ID rsa3072, created 2018-07-09 (main key ID rsa3072)

gpg: encrypted with 4096-bit RSA key, ID rsa3072, created 2018-07-09

"Harry Potter <HarryPotter@gmail.com>"

**pc@pc-Lenovo-S510:~$ ls**

add.tcl Documents examples.desktop mypk secrets\_fron\_sam Pictures secrets simple.tcl Templates Videos Desktop Downloads Music secrets\_san out.nam Public semil.py snap test.c

**pc@pc-Lenovo-S510:~$ cat secrets\_from\_san**

hello

how r u?

**CONCLUSION:**

From this experiment we learnt about new tool of linux called GnuPG tool. which is used for email security. We understood that gpg tool will use symmetric key cryptography , and public-key cryptography for ease of secure key exchange .We saw that it will create a rsa public and private key for the user. Then we created a file and encrypted that file using Gpg and also decrypted the same.