

MIT WORLD PEACE UNIVERSITY

Information and Cybersecurity
Second Year B. Tech, Semester 1

EMAIL SECURITY USING - PGP

LAB ASSIGNMENT 8

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

April 30, 2023

Contents

1 Aim	1
2 Objectives	1
3 Theory	1
3.1 PGP	1
3.2 Steps to Send an EMail using PGP	1
3.2.1 Generate a key Pair	1
3.2.2 Share public keys	1
3.2.3 Encrypt the message	1
3.2.4 Sign the message	2
3.2.5 Send the message	2
3.2.6 Decrypt the message	2
4 Platform	2
5 Input and Output	2
5.1 Generated Keys	2
6 Conclusion	5
7 FAQ	6

1 Aim

Demonstrate Email Security using - PGP or S/MIME for Confidentiality, Authenticity and Integrity.

2 Objectives

To learn authentication technique for access control

3 Theory

3.1 PGP

1. PGP (Pretty Good Privacy) is a data encryption and decryption computer program that provides cryptographic privacy and authentication for data communication. PGP is used for signing, encrypting, and decrypting texts, e-mails, files, directories, and whole disk partitions and to increase the security of e-mail communications.
2. Phil Zimmermann developed PGP in 1991. PGP and similar software follow the OpenPGP standard (RFC 4880) for encrypting and decrypting data.
3. 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.
4. PGP encryption should only be used with data that is transferred via file transfer applications that use secure connections. PGP should not be used with email applications that send and receive data in plain text. PGP encryption is not compatible with

3.2 Steps to Send an Email using PGP

3.2.1 Generate a key Pair

The first step is to generate a key pair consisting of a private key and a public key. The private key is kept secret and is used for decrypting messages that are encrypted using the corresponding public key. The public key is shared with others so they can encrypt messages that only the owner of the private key can decrypt.

3.2.2 Share public keys

In order to exchange encrypted messages, each person needs to share their public key with the other person. This can be done by sharing the key through a key server, sending the key as an email attachment, or sharing it in person.

3.2.3 Encrypt the message

Once the public keys have been exchanged, the sender can encrypt the message using the recipient's public key. The encrypted message can only be decrypted by the recipient using their private key.

3.2.4 Sign the message

The sender can optionally sign the message using their private key. This adds a digital signature to the message, which provides a way for the recipient to verify that the message was actually sent by the claimed sender, and that it has not been altered in transit.

3.2.5 Send the message

The encrypted and optionally signed message can now be sent to the recipient through email or another messaging service.

3.2.6 Decrypt the message

Upon receiving the message, the recipient can decrypt it using their private key. If the message was also signed, the recipient can verify the digital signature using the sender's public key.

These steps are shown below in Input and outputs.

4 Platform

Operating System: Arch Linux x86-64

IDEs or Text Editors Used: Visual Studio Code

5 Input and Output

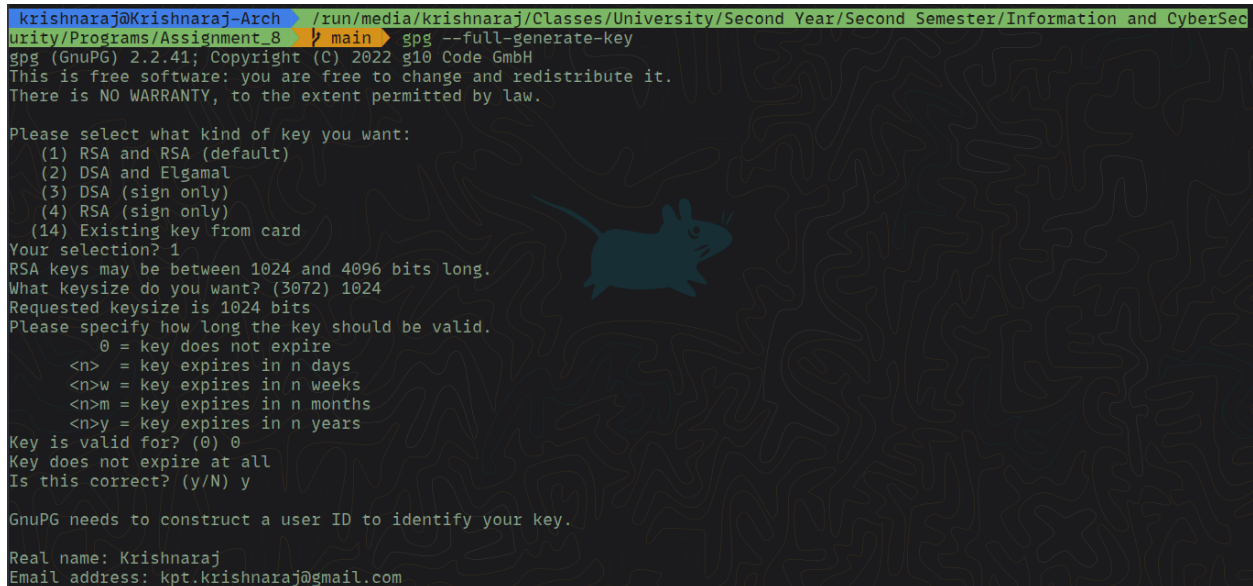
5.1 Generated Keys

```
1 -----BEGIN PGP PRIVATE KEY BLOCK-----
2
3 lQIGBGQ3mhsBBAC00HioVaOXhVb0po7Gw9/LlM4K9Lvmi0djRaQdJs8kTagJGJkP
4 yQk2Dpt0oxN3FY4/HeutrKnn/EYDnuTmglXvo7Vbessfbn9h6NQ9ZyCJmln/SAQT
5 Vz14PtaXymiii+puG7QC0rIYA4acNc2kWiGoC0kyqt1jAL0+5K8qxrGKhWARAQAB
6 /gcDAnqziXIUqkxp+emZF2o8uw7JRWCyLUedfNztqU+Jy8XcQJeSEhqobqM3p8X6
7 z0lJ9QHm6lr6BduvzulxBnkjuqPyCOUmJz2C7368uq2mNgYwb7w9tfYhI608wWVs
8 LCBA/f0mLVhAqCBhj26HNYkPYkt1PcKgYU/Qx2R0bDhMavNMwWxmE9IAkySQpQjb
9 m0tcYva53lj3jnj2LmWinvvD88P2uHvd4fXGmomqIzeRK2rADICy4z8s4o72BLVL
10 zCGITg4YLChjJttxtUlJSobApla//ckJFNdnuJbsjNColrN+J8XYPMsI/C/aTwxR
11 X21wJcwf7tVn1Ps9BXY3IGZurrbGbwg9I8V/091hb5od3J5IP3CJrs0i8CTwu7/1
12 FaKWTDSl3aIpFbqdeC1CreJF4fMBbe1WwECdAgYEZDeaGwEEALeH73HJ8lyR6pN1
13 I7pT+FNgutYL49S50XPyPh4jcQ3EqV4Sm96Xy6YBpFEsHvokWHSBrmiMe30EGKdY
14 go8g0tKBMG1sByiPSMe/ktLChWae2EV/taWkynhNwisrAonLrYCVo/FRR/PIBaT8
15 qWC8P6w4XIRUpKIi2QEhd+1sIGvbYUKUbkeadnDunDgyg/7gb/H20TXaL6XvMyzT
16 QjWlWBy+GMuL1fSCB3Kh1cRVHVsjdjj1V2c1zx0lilTo08PNV2X6NvDNI/6yfuRX
17 NeNUq//u3WJd1Ayq/2SE91bOM0LLKg5/42ZUoQetC6VEIcjL7c2/XQ9rH6vgy/BP
18 6A378Y1ZZo3Z0zvCyVh5NMhgKYi2BBgBCAAGFiEEseSJ53V8cfAaI9i3JiEHouMD
19 /UIFAmQ3mhsCGwwACgkQJiEHouMD/UImEgP8D/hMV0JB1SWERGuGqthbsslrWtKB
20 uFnGobSJuHCjRtUlzsRLcKixulMXyJiuTmtqpjEl1i2BqhpZqz6IDojFn8Q0KsWw
21 g9Hw/cFGwTPNPgujchRWZFkOqu+BBzzgl/bS1+EpbkAD7oxkJZxZI7yjQGNTTrJJ5
22 3rJ6wrgXJLQyuJs=
23 =hg+7
24 -----END PGP PRIVATE KEY BLOCK-----
```

Listing 1: "Krish Private Key"

```
1 -----BEGIN PGP PUBLIC KEY BLOCK-----
2
3 mIOEZDeoGgEEANBxkECodUnkiOouPXtisjdT0b6Z6ASGjaaYltwlNp1PjuIE1E1E
4 /U+FXB1yPnzQXYJNzZinyKALznNHPA5u1q3kfkxHx0Bs2Jr79Ly8VcFOXi621c0
5 Lw50mFDBYCotKHqRMQa4V4ovoG1toJn2RDEbYYo5zpj/cVIS4R9M+3qFABEBAAG0
6 NktyaXNobmFyYWogKGljcyBhc3NpZ25tZW50KSA8a3B0LmtyaXNobmFyYWpAZ21h
7 aWwuY29tPoj0BBMBCAA4FiEE0HV08avaij3dBuGrN2zFzRsyHgFamQ3qBoCGwMF
8 CwkIBWIGFQoJCA5CBByCAwECHgECF4AAAGkQrN2zFzRsyHgE0AQAg+kQmV7DK0X1
9 gOzyW0oDcbyLhnTodytDT2RZnNi96d+cFjRfXDB3ET26gBKVzn7b8QG5J07jSZw2
10 6noliMFlbM7JFJDP881Dr5SeSqWQZnB2Mozc0gqrlFUYmlShppoJ0wPY5y2ME8U8
11 g2u1rx3EUsu4GxyQUguy07S+u4DDL024jQRkN6gaAQAr087+003qyBmi88xSx9u
12 ktouH0so25kAR3t13tjB2hCstRZoQgGJCbKnt6kOfhRmEqYFwgUKULYvODs+GnWM
13 ah/E2shSlcSPp0dAajovHTSncOUJmU6qZDLUk78j7NMiFwf3M1vLU2KkBg6Jqw5QT
14 rpyXDIYMV2RQ3c0/ns7bMd8AEQEAAyi2BBgBCAAgFiEE0HV08avaij3dBuGrN2z
15 FzRsyHgFamQ3qBoCGwWACgkQrN2zFzRsyHgE4AP/WNEAzJuQLh/eyPNW/Cz0wkQK
16 REeu1xjOpFGdXfPytoe1GT+xZ6oDd0WHPzMb3oa7NDC4DbzbRw7AfLJj2Qo4PdYG
17 vmEHmlON7NY4Wv5W73bxiR8HJGThL6SS4TJXF/etW/Jxs/LYaEnifa77quKeX0Sh
18 hI+UmhljipkecyXakA4=
19 =Qz5/
20 -----END PGP PUBLIC KEY BLOCK-----
```

Listing 2: "Krish Public Key"



```
krishnaraj@Krishnaraj-Arch: /run/media/krishnaraj/Classes/University/Second Year/Second Semester/Information and CyberSec
urity/Programs/Assignment_8 $ main gpg --full-generate-key
gpg (GnuPG) 2.2.41; Copyright (C) 2022 g10 Code GmbH
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
  (1) RSA and RSA (default)
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)
  (14) Existing key from card
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (3072) 1024
Requested keysize is 1024 bits
Please specify how long the key should be valid.
  0 = key does not expire
  <n> = key expires in n days
  <n>w = key expires in n weeks
  <n>m = key expires in n months
  <n>y = key expires in n years
Key is valid for? (0) 0
Key does not expire at all
Is this correct? (y/N) y

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

Real name: Krishnaraj
Email address: kpt.krishnaraj@gmail.com
```

Figure 1: Generating Key Pairs

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

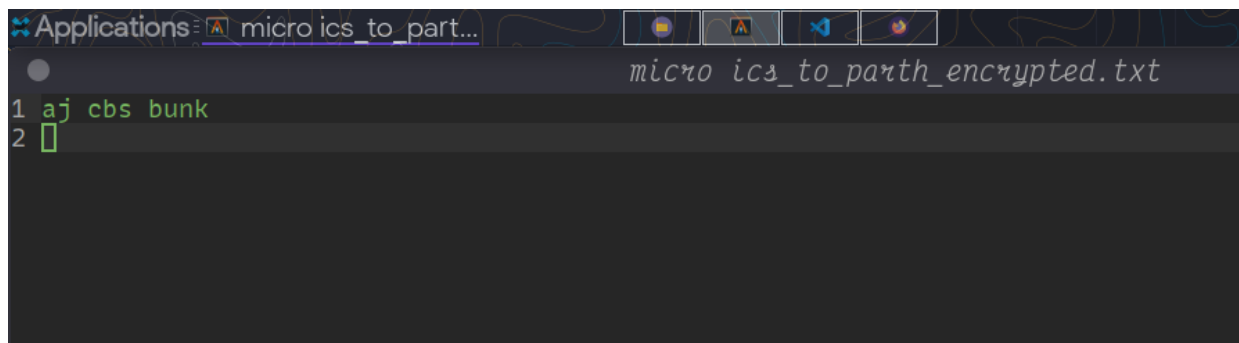
Real name: Krishnaraj
Email address: kpt.krishnaraj@gmail.com
Comment: ics assignment
You selected this USER-ID:
    "Krishnaraj (ics assignment) <kpt.krishnaraj@gmail.com>"

Change (N)ame, (C)omment, (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: revocation certificate stored as '/home/krishnaraj/.gnupg/openpgp-revocs.d/D341D53BC6AF6A28F7741B86ACDDB317346CC878.rev'
public and secret key created and signed.

pub  rsa1024 2023-04-13 [SC]
    D341D53BC6AF6A28F7741B86ACDDB317346CC878
uid                               Krishnaraj (ics assignment) <kpt.krishnaraj@gmail.com>
sub  rsa1024 2023-04-13 [E]

krishnaraj@Krishnaraj-Arch: /run/media/krishnaraj/Classes/University/Second Year/Second Semester/Information and CyberSecurity/Programs/Assignment_8 → main
```

Figure 2: Generating Key Pairs continued



The screenshot shows a terminal window titled "Applications: micro ics_to_part...". The terminal is running the "micro" text editor, editing a file named "ics_to_parth_encrypted.txt". The editor shows two lines of text: "1 aj cbs bunk" and "2". The cursor is at the end of the second line.

Figure 3: Secret Message to Parth - Recipient - "Aj CBS Bunk"

```
krishnaraj@Krishnaraj-Arch: /run/media/krishnaraj/Classes/University/Second Year/Second Semester/Information and CyberSec
urity/Programs/Assignment_8 $ main $ micro ics_to_parth_encrypted.txt
krishnaraj@Krishnaraj-Arch: /run/media/krishnaraj/Classes/University/Second Year/Second Semester/Information and CyberSec
urity/Programs/Assignment_8 $ main $ gpg --armor --export --output krish_public_key.txt kpt.krishnaraj@gmail.com
File 'krish_public_key.txt' exists. Overwrite? (y/N) y
krishnaraj@Krishnaraj-Arch: /run/media/krishnaraj/Classes/University/Second Year/Second Semester/Information and CyberSec
urity/Programs/Assignment_8 $ main $ echo sending public key to parth, along with encrypted file
sending public key to parth, along with encrypted file
krishnaraj@Krishnaraj-Arch: /run/media/krishnaraj/Classes/University/Second Year/Second Semester/Information and CyberSec
urity/Programs/Assignment_8 $ main $ gpg --encrypt --sign --recipient 1032210846@mitwpu.edu.in ics_to_parth_encrypted.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: 6587AE74FD6336BC: There is no assurance this key belongs to the named user

sub rsa1024/6587AE74FD6336BC 2023-04-13 Parth (study material for krishnaraj) <1032210846@mitwpu.edu.in>
Primary key fingerprint: 51C8 CD14 8CF1 DC1A 4511 A8AC 6F06 3B23 E66F 9B7F
Subkey fingerprint: 29F5 6B97 8119 0CE2 BE65 D4F3 6587 AE74 FD63 36BC

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
File 'ics_to_parth_encrypted.txt.gpg' exists. Overwrite? (y/N) y
krishnaraj@Krishnaraj-Arch: /run/media/krishnaraj/Classes/University/Second Year/Second Semester/Information and CyberSec
urity/Programs/Assignment_8 $ main $
```

Figure 4: Signing Message to Send Parth using Parth's Public Key

```
krishnaraj@Krishnaraj-Arch: /run/media/krishnaraj/Classes/University/Second Year/Second Semester/Information and CyberSec
urity/Programs/Assignment_8 $ main $ gpg --decrypt ics_confidential_file.txt\2\).gpg
gpg: encrypted with 1024-bit RSA key, ID 9CC98A09B693ED66, created 2023-04-13
"Krishnaraj (highly secret stuff bro) <1032210888@mitwpu.edu.in>"
ajj sab bunk
gpg: Signature made Thu Apr 13 12:18:08 2023 IST
gpg: using RSA key 51C8CD148CF1DC1A4511A8AC6F063B23E66F9B7F
gpg: Good signature from "Parth (study material for krishnaraj) <1032210846@mitwpu.edu.in>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg: There is no indication that the signature belongs to the owner.
Primary key fingerprint: 51C8 CD14 8CF1 DC1A 4511 A8AC 6F06 3B23 E66F 9B7F
krishnaraj@Krishnaraj-Arch: /run/media/krishnaraj/Classes/University/Second Year/Second Semester/Information and CyberSec
urity/Programs/Assignment_8 $ main $
```

Figure 5: Decrypting Message from Parth using his Public key - "Ajj Sab Bunk"

6 Conclusion

Thus, we have successfully implemented Email Security using - PGP or S/MIME for Confidentiality, Authenticity and Integrity.

7 FAQ

1. How email security is provided through PGP?

- (a) PGP (Pretty Good Privacy) provides email security through a combination of encryption, digital signatures, and compression. When a user sends an email using PGP, the message is encrypted using a symmetric key algorithm.
- (b) The symmetric key is then encrypted using the recipient's public key, which is obtained from a key server or a public key directory. The encrypted message and the encrypted symmetric key are then sent to the recipient, who can decrypt the message using their private key.
- (c) PGP also allows users to sign their emails digitally using their private key. The digital signature provides a way for the recipient to verify that the email was actually sent by the claimed sender, and that it has not been altered in transit.
- (d) In addition, PGP can compress the message before encryption, which can reduce the size of the message and make it easier to send over a slow or unreliable connection.

2. What type of encryption is PGP?

PGP uses a combination of symmetric and asymmetric encryption. The symmetric encryption algorithm is used to encrypt the message itself, while the asymmetric encryption algorithm is used to encrypt the symmetric key.

The symmetric encryption algorithm used in PGP is typically AES (Advanced Encryption Standard), which is a widely used and highly secure algorithm. The asymmetric encryption algorithm used in PGP is typically RSA (Rivest–Shamir–Adleman), which is also widely used and highly secure.

3. What is the key size allowed in PGP

PGP supports a wide range of key sizes, from 512 bits to 4096 bits. The key size determines the level of security provided by the encryption algorithm.

In general, larger key sizes provide stronger security, but they also require more processing power to encrypt and decrypt the data. For most purposes, a key size of 2048 bits is considered to be sufficient, but some applications may require larger key sizes for enhanced security.