Lab 4: Public Key Cryptography and Digital Signature

1 Overview

The objective of this lab is to (i) learn and practice how to use GPG to encrypt/decrypt files with asymmetric algorithms and (ii) learn how to sign/verify a file using digital signatures. During the lab you will generate your own key-pair and learn how to use it. **You will be working in groups and this lab will form Lab5 assignment which is due in Week-12.**

2 Lab Environment

Virtual Machine: Ubuntu can be downloaded from
https://drive.google.com/open?id=0BxtCKJljOE-NUnY0VGMwaDdyRTA

GPG: GnuPG, also known as GPG, is already installed in our VM. GPG is a complete and free implementation of the OpenPGP standard as defined by RFC4880 (also known as PGP). GnuPG allows to encrypt and sign your data and communication, features a versatile key management system as well as access modules for all kinds of public key directories. GPG is a command line tool with features for easy integration with other applications.

3 Lab Tasks

3.1 Task 1: Public Key Encryption in GPG

In this task, every group will generate a key pair, share their public key with another lab group and encrypt/decrypt files.

3.1.1 Generating Key-pair

Head to the terminal and write:

```
$ gpg --gen-key
```

GPG will then prompts you through the process, giving options for each question.

- The type of key you want: RSA and RSA,
- the key length: 1024,
- the expiration time, default is ”key does not expire”,
- your real name: Your name,
- your email address: your@address.com
- a comment: ”My student mail”,
- and a passphrase used to protect you key.

The keys are generated in a hidden directory .gnupg in your home folder. You can view it by navigating to your home folder and type: `ls -la`. Please notice the permission of file `.gnupg`.

3.1.2 The revocation certificate

You will need a way to invalidate the key if it gets compromised. Therefore, you shall now generate a revocation certificate. Run the following command in terminal, this will ask a number of questions before saving the certificate to `revoke.asc` file.

```
$ gpg --output revoke.asc --gen-revoke your@address.com
```

Generating the revocation key should be the second next thing to do after generating a key-pair, because if they key actually gets compromised, then it may be too late. The revocation certificate should be stored where others have no access, since anyone can publish it and render your key useless.

3.1.3 Exchanging Keys

To communicate with others you must exchange public keys. To get hold on your public key you export it from your keyring using the command-line:

```
$ gpg --export your@address.com > your_public.gpg
```

The key is exported in binary format, which is not very well suited as a visual representation. By also giving the option `--armor`, `gpg` will use a binary-to-ASCII encoding to output a more readable file.

```
$ gpg --export --armor your@address.com
$ gpg --export --armor your@address.com > your_public.gpg
```

Please share `your-public.gpg` file with other lab group and request them to send their's. To import other lab group's key and view the keys available in your keyring:

```
$ gpg --import your_friend_public.gpg
$ gpg --list-keys
```

Once you have imported a key it should be validated. This is done by generating a fingerprint from the imported key and verify that to be the same as the one generated by the owner.

```
$ gpg --fingerprint friend@address.com
```

If all checks out, then you can be confident that you have a correct copy of the key, and you certify that by signing the copy with your own key. You should be very careful to always verify the keys fingerprint with the owner before you sign.

```
$ gpg --sign-key friend@address.com
```

3.1.4 Encryption and Decryption

Please create a text file `plain.txt`, if you want to encrypt the file so that only for `friend@address` to read, then type:

```
$ gpg --encrypt --output cipher.txt --recipient \
  friend@address.com plain.txt
```

Which will create the encrypted `cipher.txt`. Please share this file through email to other lab group. The file can be decrypted by typing:

```
$ gpg --output plain-dec.txt --decrypt friend-cipher.txt
```

You should now transmit and receive messages securely with another lab group.

3.2 Task 2: Signing and authenticating

To sign a text file:

```
$ gpg --sign plain.txt
```

will create a compressed file of the `plain.txt` together with the signature called `plain.txt.gpg`. To verify signature:

```
$ gpg --verify plain.txt.gpg
```

and

```
$ gpg --output plain.txt --decrypt plain.txt.gpg
```

to verify and retrieve the file.

Please take screenshot after every step and submit in your Lab5 assignment.