

Introduction to PGP encryption

This hands-on lab session aims to introduce you to PGP (Pretty Good Privacy), widely used encryption software for securing email communication and data files. Through practical exercises, you will learn how to generate PGP key pairs, encrypt and decrypt messages, sign and verify files, and exchange encrypted emails. This lab will provide all the essential skills for using PGP for secure communication and data protection.

Objectives:

- Understand the principles of PGP encryption and digital signatures.
- Generate PGP key pairs for encryption and signing.
- Encrypt and decrypt messages using PGP.
- Sign and verify files using PGP digital signatures.
- Exchange encrypted messages with classmates.

Generating PGP Key Pairs to begin GPG tool to encrypt all the communication we want.

Use your VM to install `gnupg` with the appropriate command (if not already installed). On Debian that is:

```
sudo apt-get install gnupg
```

First, create a keypair with the command:

```
gpg --generate-key
```

Tip: Don't forget the passphrase, use a strong one!

You can send your public key by exporting it through the GPG application:

```
$ gpg --output ~/mypub.key --armor --export youremail@mail.com
```

You can send the key by email or any method of your choice.

How to import locally foreign public keys

We need to accept other users' public keys if we want to communicate with them. When you receive others' public keys, import them using the `import` parameter.

Tip: There are other options to share your public key, such as using a public key server. Here's how to verify other keys.

While it's generally safe to distribute your public keys, caution should be exercised when obtaining keys from other users. Therefore, it's crucial to verify the identity of others before importing their keys and initiating communication. A quick method to accomplish this is by comparing the fingerprints derived from those keys. Use the following command:

```
$ gpg --fingerprint youremail@mail.com
```

This will result in a more concise string of numbers, making it easier to compare. We can then verify this string either with the individual themselves or with another party who has access to the person in question.

Now sign the keys

By signing each other's keys, we essentially indicate trust in the keys we possess and confirm the identity of the individual associated with that key.

```
$ gpg --sign-key youremail@mail.com
```

To ensure that the person whose key you're signing benefits from your trusted relationship, send them back the signed key. You can accomplish this by typing:

```
$ gpg --output ~/signed.key --export --armor youremail@mail.com
```

Upon receiving this newly signed key, they can import it, incorporating the signing information you've generated into their GPG database. They can achieve this by using the `import` parameter.

Now that you have established a secure channel to communicate with trusted ends, you can send and receive encrypted messages with each other!

After sharing your key with the other party, you can now send and receive encrypted messages. You can encrypt your messages with the following command:

```
$ gpg --encrypt --sign --armor -r other_person@mail.com file_name
```

This command encrypts the message and signs it with your private key to ensure that the message originates from you. The output will be an encrypted file with the extension `.asc`. You can use the `cat` command to view the file.

Keep in mind that this message will be encrypted using the recipient's public key, making it unreadable to you (Unless you somehow obtain the recipient's private key!).

Tip: You can modify the command above to generate a message only you can read, using your private key. Think about how.

Decrypt received messages

Upon message receipt, you would be able to read with the `gpg` command and the `decrypt` parameter.

Tip: In case you have a raw stream text message, paste the message after typing `gpg` with no arguments. Then press `CTRL+D` to end the message.

If you've completed all the above steps correctly, you can now communicate securely with different individuals. You can send your public key to multiple recipients and import multiple keys as well. This is an essential step to keep your messages private and protect your sensitive information from eavesdropping attacks.

Import other people's public keys from a public key server

An alternative method to share your GPG keys is through a public keyserver. Anyone can upload a key to a public server, making it accessible to others who may need to communicate securely with you.

Export your public key using:

```
$ gpg --armor --export email
```

Send your public key to a public server using the `--send-key` parameter. By default, your key is sent to <https://keys.openpgp.org/>, or you can specify a different key server using the `-keyserver` option.

Before importing a user's key, verify its validity with the command `gpg --list-sigs user_id` where the `user_id` is for example the email address you want to communicate with. To find a key on a public keyserver, use the `--search` parameter.

Import the key you want with the `--import` parameter. You can once again verify the sender's identity using the `--fingerprint` parameter and checking its validity via a second channel. Then, similarly, sign the key obtained from the public key server.

Here you can find some key IDs of people you may certainly know:

- **413109AF27CBFBF9,**
- **9A88F479F6D1BBBA,**
- **1C29680110FA7E87**

These keys are on the <http://keyserver.ubuntu.com//> public keyserver.

- Use the `gpg` command with the `--search` parameter to find out, for each key, the associated email address or User ID (username).
- Use the `gpg` command with the `--recv-keys` parameter if you decide to import a key into your keyring.
- Use the `gpg` command with the `--fingerprint` option to verify the authenticity of the keys.
- Now, can you try searching based on the email addresses?
- Can you find Joseph's old expired key?