# GnuPG Summary

Blake McBride

March 28, 2023

# Contents

**1 Introduction and Setup** **1**
1.1 Setup . . . . . 1
1.1.1 Creating Your Public / Private Keys . . . . . 1
1.1.2 Importing Other People's Public Keys . . . . . 2
1.1.3 Other Key Operations . . . . . 3

**2 Encrypting and Decrypting Files** **5**
2.1 Public/Private Key Encryption . . . . . 5
2.2 File Decryption . . . . . 5
2.3 Conventional/Password Encryption . . . . . 6

**3 Signing Files** **7**
3.1 Signing A File . . . . . 7
3.2 Verifying A File . . . . . 7
3.3 Extracting A File . . . . . 8

[Blank Page]

# Chapter 1

# Introduction and Setup

*GnuPG* (gpg) is a freely available file encryption program which can encrypt files with either public/private key encryption or conventional/password encryption. Public/private key encryption is stronger for two reasons. First, it can only be decrypted by a set of intended recipients. Conventionally encrypted files can be decrypted by anyone who knows (or can guess) the password. Second, public/private key pairs are very long and very random when compared to passwords used in conventional encryption. This makes public/private key encryption much more secure.

## 1.1 Setup

If your computer doesn't already have *gpg* installed, you may download it from:

```
https://gnupg.org
```

### 1.1.1 Creating Your Public / Private Keys

Once installed, you need to generate your own public/private key pair as follows.

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

Use the defaults as much as possible. Create it to never expire. Make sure you remember your password.

This process creates a private and public key for you as well as other system files. On Unix-like systems, all of these files are stored in the */.gnupg* directory. A secure backup of this directory should be kept!

Your public key must be given to all people you wish to exchange encrypted files with. Your private key is private. You should never give it to anyone. Also, your private key is stored on your computer in a place where your OS stores your user specific files. This file must be protected.

In what follows, all instances of *youremail* should be replaced by your email address. All instances of *theiremail* should be replaced by the email address of the other person you are communicating with.

Once you have created your public/private key you need to put a copy of your public key in a file so you can share it with others. A file with your public key can be created as follows:

```
gpg --export --armor youremail >youremail.key
```

### 1.1.2 Importing Other People's Public Keys

Public keys that others send you can be imported as follows:

```
gpg --import theiremail.key
```

Once their public key has been imported, it must be validated. This assures that the key sent to you was in fact sent by the person you think it was. This can be done as follows:

```
gpg --sign-key theiremail
```

This will display their public key fingerprint and ask you if you accept it. You should call the person on the phone or in person and validate that finger print. They can display their fingerprint with the following command.

```
gpg --fingerprint
```

If the fingerprints match then you can accept it on your end.

### 1.1.3 Other Key Operations

You can list the people in your public key ring as follows.

```
gpg  --list-keys
```

You can delete someone from your public key ring as follows.

```
gpg  --delete-key  theiremail
```

# Chapter 2

# Encrypting and Decrypting Files

## 2.1 Public/Private Key Encryption

A file can be encrypted with public/private keys in such a way that only the intended recipients can decrypt the file with the following command.

```
gpg  -e  -r  youremail  -r  theiremail  myfile
```

File *myfile* will be encrypted such that only *youremail* and *theiremail* can decrypt it. You can add as many *"-r email"* options as people who you want to allow to decrypt the file. It is important to include yourself as a recipient so that, if needed, you can decrypt the file.

This command will create a file named *myfile.gpg* which is the encrypted version of the file.

## 2.2 File Decryption

A file may be decrypted with the following command.

```
gpg  -q  myfile.gpg
```

This will decrypt the file into a file with the same name but with the *.gpg* file extension removed.

## 2.3 Conventional/Password Encryption

A file may be encrypted without the use of the public / private keys as follows.

```
gpg -c myfile
```

*gpg* will prompt you for a pass-phrase. Anyone with the pass-phrase will be able to decrypt the file. The encryption is only as good as the pass-phrase therefore this encryption method is not as good as public / private key encryption and should be avoided.

Files encrypted conventionally are decrypted in the same fashion as public / private key encrypted files.

# Chapter 3

# Signing Files

*gpg* has the ability to electronically sign a file. What this means is that you can be assured of who the file came from and that it has not been modified.

## 3.1 Signing A File

A file may be signed with the following command:

```
gpg  -s  myfile
```

This will create a file names *myfile.gpg* which contains the file and its signature. *myfile* may be deleted.

## 3.2 Verifying A File

A file may be verified with the following command:

```
gpg  --verify  myfile.gpg
```

## 3.3 Extracting A File

Lastly, the original file may be extracted using:

```
gpg  -q  myfile.gpg
```