### GPG Basic Usage

#### Creating keys

gpg --gen-key

It will ask lots of questions to create the key. You can use the default values. Remember your (optional) passphrase.

#### Exporting keys

gpg --armour --export "Ciprian <ciprian@example.com>" > pubkey.asc

Your public key is pubkey.asc. You can check the current keys present using:

gpg --list-keys

A sample output will be:

~/.gnupg/pubring.gpg------------------------------

sec 2048R/C0FE3AME 2016-01-13

uid ciprian

ssb 2048R/DD2EAP22 2016-01-13

The keyID is C0FE3AME.

#### Submitting keys to a keyserver

To submit keys to a keyserver, say, pgp.mit.edu, do:

gpg --keyserver pgp.mit.edu --send-key C0FE3AME

Generally, when using GPG, you want others to have the ability to verify your signatures or encrypt data to you. In order to do so, they need your public key. To help them obtain it conveniently, you can put it on a public keyserver.

You can set the keyserver to use in the configuration file ~/.gnupg/gpg.conf with the keyserver directive, or via the command-line option gpg --keyserver; both take an URL as an argument, such as hkp://subkeys.pgp.net. However, all of the major keyservers communicate with each other and synchronize keys, so you usually don’t need to change the default.

To send your key to a keyserver, you need to know your key ID. You can print the information on all keys you have the private key for by running gpg --list-secret-keys. This will generate output similar to the following:

~/.gnupg/secring.gpg-----------------------------

sec 2048R/C0FE3AME 2016-01-13

uid ciprian

ssb 2048R/DD2EAP22 2016-01-13

From this, you can see my primary key ID, C0FE3AME. Now that you know your key ID, you can send your public key to the default keyserver with the gpg --send-keys option:

$ gpg --send-keys C0FE3AME

gpg: sending key C0FE3AME to hkp server subkeys.pgp.net

Keyservers distribute public keys to anyone who requests them. Once you have sent your key to a keyserver, others can request your key using the gpg --recv-keys option, like gpg --recv-keys C0FE3AME. To refresh all your keys from a keyserver, to obtain new signatures, new UIDs, or key revocations, use gpg --refresh-keys; you should do this regularly.

#### Searching for keys

You can search for keys using:

gpg --keyserver pgp.mit.edu --search-keys "Ciprian"

#### Importing keys

To import keys to your pubring, you can do:

gpg --import whoispubkey.asc

#### Signing documents

To sign a document to send it to say, [ciprian@example.com](mailto:ciprian@example.com), use the --encrypt option. You must have Ciprian’s public key in your pubring.

gpg --output doc.gpg --encrypt --recipient \ ciprian@example.com document

As Ciprian if you want to decrypt the above message, you can do:

gpg --output document --decrypt doc.gpg

It will ask for your passphrase.

#### Clearsign

You can also clearsign the document to be sent, via e-mail, for example, use:

gpg --clearsign document

The document contents will be embedded between the PGP signed message, as shown below:

-----BEGIN PGP SIGNED MESSAGE-----Hash: SHA1

[----document-content-----]

-----BEGIN PGP SIGNATURE-----Version: GnuPG v0.9.7 (GNU/Linux)Comment: For info see https://www.gnupg.org/

iaYEA3ECAbYFA2dY3Qo4Cgk2J916UL31dqz4IwC5Q7wP6j/i8lhbcwSK4rLyQB1

oCoAoOwqpaqEfr4eOksqHeLE/r8/Ra2k=y3k2-----END PGP SIGNATURE-----

### Examples:

### Encrypt a file and send it to a friend

**1. Import your friend’s public key**

When you want to send a secret file to your friend, the first thing to do is to import your friend’s public key. You can import the key from a key server if he/she has previously exported their public key to a key server.

You can import by using any one of the below methods. Provide your friend’s key ID or email ID or real name to import the keys correctly.

gpg --search-keys --keyserver keyserver.ubuntu.com 'key ID here'

or

gpg --search-keys --keyserver keyserver.ubuntu.com 'email ID here'

or

gpg --search-keys --keyserver keyserver.ubuntu.com 'real name here'

Note that some friends might not have an email or a real name, only a key ID.

If your friend has emailed you his/her public key, then you can import that key by using the following command:

gpg --import myfriends\_pub\_key.gpg

**2. Verify the imported key**

You can verify whether you have successfully imported your friend’s public key using the --list-keys option

gpg --list-keys

~/.gnupg/pubring.gpg-----------------------------------

pub 2048R/A734DE7D 2016-11-12

uid myselfsub 2048R/96A8EFAB 2016-11-12

pub 2048R/FBC744A8 2016-12-13

uid friendsub 2048R/88EFF5E5 2016-12-13

Now, I have my friend’s public key imported.

**3. Encrypt a secret file using your friend’s public key**

Now that you have the public key of your friend, you can send him a file, which is encrypted using his/her key, so only your friend, can decrypt it.

gpg --encrypt --recipient friend file.txt

gpg: 88EFF5E5: There is no assurance this key belongs to the named user

pub 2048R/88EFF5E5 2012-12-03 friend

Primary key fingerprint: FF32 7764 A0AE 1F85 AC4B CF17 8AED B212 FB27 4FA8

Subkey fingerprint: D6A5 7137 77F8 6845 2F86 765C EDED DD85 88EF 55ED

It is NOT certain that the key belongs to the person namedin 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

It will then create a file named file.txt.pgp which, when opened using any editor, will have some binary data.

If you don’t want to send binary content, or there are issues in sending binary, you can use the --armor option which creates an ASCII file as shown below:

gpg --encrypt --armor --recipient friend file.txt

**4. Decrypt a file**

In order to view the content of the file, your friend needs to decrypt the file. Since decryption operation will be performed using your friend’s private key, it will (optionally) ask for the passphrase provided by your friend while creating keys.

gpg --decrypt file.txt.gpg > secret.txt

Now the file secret.txt contains the actual text decrypted.

5. Send an encrypted file to multiple recipient

You can also send a file to multiple recipients by using the --recipient or -r option.

$ gpg -r friend -r colleague -r mate --encrypt file.txt

Once the above command is given, gpg will use the public key of all the recipients to encrypt the data in such a way that any one of their private keys can decrypt the data.