What is GPG?

GPG (GNU Privacy Guard) is a free and open-source implementation of the PGP software suite

PGP is made by Symantec (sorta)

What is PGP?

PGP (Pretty Good Privacy) is a proprietary cryptographic suite developed by Phil Zimmerman.

Released in 1991.

Wait, proprietary?

- Yup, that's how it started
- However, nowadays PGP follows RFC 4880, which is the OpenPGP Standard.
- However, when did it become an open standard?

Munitions

- As you may or may not be aware, exporting strong crypto (>40bits) were "munitions"
- Phil Zimmerman became the target of a federal criminal investigation

His response

- To challenge the regulations, Zimmerman released the source code of PGP in a physical book
- The book was distributed by MIT Press
- Export of books are protected by the first amendemnt, techincally not exporting software...

Late 90s—2010

- PGP went worldwide as an open application, PGP Inc was made in the mid—90s
- PGP Inc. was bought by Network Associates, Inc. (McAfee) in Dec. 1997
- PGP Corp. was made and bought most of the PGP assets from NAI in 2002.
- PGP Corp. was bought by Symantec in 2010.
- Intel acquired McAfee in 2010 as well.
- Broadcom acquired Symantec's Enterprise security software division in 2019.

OpenPGP

- When PGP Inc. was developing PGP 3, they needed no licensing issues
- Viacrypt's RSA use was being challenged by RSADSI.
- Unencumbered PGP was used internally and drafted for the IETF in Jul. 1997

GnuPG

- Released in Sep. 1999
- Completely FOSS implementation of OpenPGP Standard

Concepts surrounding Asymmetric Encryption

- You generate a **key—pair** (Public key and private key)
- You can freely distribute the public key
- You **MUST** keep the private key secret!

Concepts surrounding GPG

- You generate a key—pair using any supported algorithm
- You can delegate certain permissions to **sub—keys**
  - This makes key management safer *and* easier, in the long run
- This key—pair is considered an *identity*.
  - This can be your IRL identity or can be an online pseudonym/identity

Other things

- Keys can store some amount of arbitrary data
  - Usually this is used to either store a comment, or a picture of some kind

How to generate a key-pair

- Make sure you have GnuPG installed
- If you want to be extra safe, do this entire process on an air-gapped system!

```
export GNUPGHOME=$(mktemp -d)
# Optional, generate a random password with uppercase and nums for master key
tr -dc '[:upper:][:digit:]' < /dev/urandom | fold -w20 | head -n1
```

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

- Choose an algorithm (I personally use ECC -> Ed25519)
- Make sure to "Choose your own capabilities"
- Toggle all actions to be disabled. You should only have "Certify" enabled
- Fill out all the information. When done, copy the Key ID.
- Run `export KEYID=<your-id-here>`

Generate the actual keys

```
gpg --expert --edit-key $KEYID
```

- Add three sub-keys, each with an expiration of 1 year
- Add extra UIDs if applicable, make sure to `trust` them all
- One for signing only, One for Encryption only, One for Authentication only
- Make sure to run `save` when done

Get the keys for use

```
gpg -a --export-secret-keys $KEYID > $GNUPGHOME/mastersub.key
gpg -a --export-secret-subkeys $KEYID > $GNUPGHOME/sub.key
gpg --output $GNUPGHOME/revoke.asc --gen-revoke $KEYID
gpg -a --export $KEYID > $GNUPGHOME/pub.key
# Get some backups now, install p7zip
7z a -p master_backup.7z $GNUPGHOME/mastersub.key $GNUPGHOME/sub.key \
$GNUPGHOME/revoke.asc $GNUPGHOME/pub.key
```

- Now you have a Primary key, three sub keys, a public key, and a revocation cert.
- If not using a yubikey, put your public and secret ***SUB***keys on a non-volatile directory. Move your encrypted archive there as well
- Once you are sure you have everything, `rm -rf $GNUPGHOME` or even `shred $GNUPGHOME`
- Re-launch your terminal to reset `GNUPGHOME`, and import your public key and secret sub-keys
- You should be done!

Yubikey specific

- If using a Yubikey, no point in putting secrets on your main computer at all!
- Plug it in, run `gpg --card-edit`
- Run `passwd` and set the various PINs. Default admin PIN is `12345678`
- Set `name`, `lang`, and `login` based on your PGP identity.
- `quit`

Moving the keys to Yubikey

This is irreversable!!! Make sure you have your backups before you do this.

```
gpg --edit-key $KEYID
key 1
keytocard
key 1 # This will deselect it
key 2
keytocard
...
```

- Now you just need to set your main environment to use these keys.
- Re-launch your terminal after shredding the temp GPGHOME.
- Import your public key only!
- Run `gpg --card-status` and the stubs for your secret keys on the Yubikey will be generated.

Next Steps

- Enable gpg-agent to allow SSH authentication using your GPG key.
- Once a year, extend the expiration on your sub-keys (proves ownership of secrets)

Enable SSH Auth

bashrc:

```
export GPG_TTY="$(tty)"
export SSH_AUTH_SOCK="/run/user/$UID/gnupg/S.gpg-agent.ssh"
gpg-connect-agent updatestartuptty /bye > /dev/null
```

~/.ssh/config:

```
Match host * exec "gpg-connect-agent UPDATESTARTUPTTY /bye"
```

~/.gnupg/gpg-agent.conf:

```
pinentry-program <path to pinentry>
default-cache-ttl 600
max-cache-ttl 7200
default-cache-ttl-ssh 1800
max-cache-ttl-ssh 7200
```

How to rotate expiration

- Re-do steps from above to make a temporary GNUPGHOME.
- Instead of making a new key, decrypt your backup files and import your public/private master keys.
- Select each key, run `expire` and set a new expiration date.
- Re-do the export of the secret keys, subkeys, rev. cert, and public key, re-install on main system/yubikey and re-upload to keyservers.

Questions?