

# GnuPG

Danny Robson

# Caveats

- I am not a cryptographer...
- but I did do 1 semester of 'computer security'...
- over 10 years ago...
- take this how you will...

# tl;dr

HOW TO USE PGP TO VERIFY  
THAT AN EMAIL IS AUTHENTIC:

LOOK FOR THIS  
TEXT AT THE TOP:



# What?

GnuPG is a command line only tool for:

- Signing
- Encrypting
- Identity management

# Key ideas

- Encryption
  - Symmetric
  - Asymmetric/public-key
- Hashing
- Signing

# Why?

- Privacy
- Trust

# Why?



# Why?





# Why?

- **From:** Danny Robson <[danny@nerdcruft.net](mailto:danny@nerdcruft.net)>  
**To:** [mlug-au@googlegroups.com](mailto:mlug-au@googlegroups.com)  
**Subject:** Re: [MLUG] Next meeting - 31st July Monday  
**Date:** Sun, 30 Jul 2017 18:49:37 +1000  
**Security:** GPG signed


On Thu, 27 Jul 2017 17:29:43 +1000  
Michael Pope <[map7777@gmail.com](mailto:map7777@gmail.com)> wrote:

> GPG/PGP key signing session & basics Danny Robson

Just a quick reminder that if you want to get your key signed at the meeting then you'll need to bring photo ID, and a copy of your full fingerprint (on paper is fine, that's how I'll be doing it).

I'll be live demoing some of the functionality, so you can follow along with your own computer if that's easier (though ID will still be required).

--  
Danny Robson

 Valid signature (Daniel Robson <[danny@nerdcruft.net](mailto:danny@nerdcruft.net)>)

# Keys

- ID
- Real name
- Email
- Comment

# Identities

Each identity includes:

- 1 primary key
- 0..n subkeys

# How to start

- Create your:
  - Primary key
  - Revocation certificate
  - Subkeys
- Upload your keys

# Primary Key

`gpg --gen-key`

or

`gpg2 --full-generate-key`

# Revocation certificate

Allows you to say 'don't trust this key anymore'.

```
gpg --gen-revoke --armour --output=  
  <path> <email|short|long>
```

- Do this. Really. Right now.
- Store it somewhere safe.

# Subkey

- `gpg --edit-key <email|short|long>`
  - `addkey`

# Encrypting

You can now email passwords to friends.

- `gpg --encrypt [--sign] --recipient=  
 <name> <filename>`
- `gpg --decrypt <filename>`



# Signing

You can prove a file hasn't been tampered with.

```
gpg --armour --detach-sign path
```

```
gpg --verify path.asc path
```

# Making it easy

- Pretty much every mail client has a GPG plugin
  - Thunderbird (enigmail), claws, evolution
  - Even Outlook

# How do we trust someone?

That's good, but the key creation process feels insecure...

```
cat >foo <<EOF
Key-Type: DSA
Key-Length: 1024
Subkey-Type: ELG-E
Subkey-Length: 1024
Name-Real: Donald Trump
Name-Comment: Totally legitimate
Name-Email: the.prez@whitehouse.gov
Expire-Date: 0
Passphrase: hunter2
%commit
EOF
gpg --batch --generate-key foo
```

# Web-of-trust

- You trust Alice, Bob, Dan, Frank
- Alice, Bob, Dan, and Frank vouch for me.
- You can *probably* trust me.

# Key-signing

1. Get your friends key
2. Check your friends key
3. Sign your friends key
4. Broadcast the signed key\*
5. Update your foreign signed keys

# Key-signing

- `gpg --recv-keys <short|long|email>`
- `gpg --fingerprint <short|long|email>`
- `gpg --sign-key <long>`
- `gpg --send-keys <long>`
- `gpg --update-keys`

# Signing Protocol

1. Meetup verification
  1. Fingerprints
  2. Identification
2. Post-meetup verification
  1. Fetch the key
  2. Verify fingerprint, name, email
  3. Confirm the email

# Email confirmation

1. Locally sign the key
2. Export the key
3. Encrypt-and-sign the exported key
4. Email to key owner
5. Owner imports and broadcasts to keyserver



# Email confirmation - Me

- `gpg --sign-key <id>`
- `gpg --export --armour <id>`
- `gpg --encrypt <id> --sign --armour  
<id> --output <id>-signedby-<me>.asc`

# Email confirmation - Them

- `gpg --import <me>-signedby-<them>.asc`
- `gpg --send-key <them>`

# Links

<https://wiki.debian.org/Subkeys>

<https://alexcabal.com/creating-the-perfect-gpg-keypair/>

<https://security.stackexchange.com/questions/31594/what-is-a-good-general-purpose-gnupg-key-setup>