APRICOT2015 Security Workshop:

Pretty Good Privacy (PGP)

Sheryl Hermoso, APNIC
sheryl@apnic.net

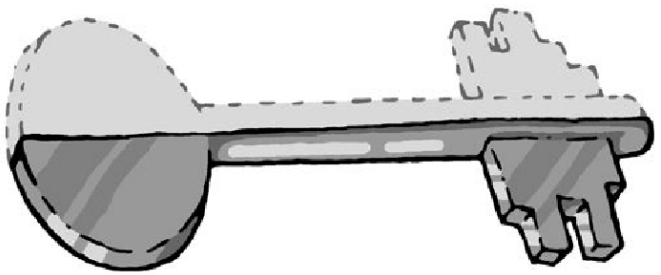
PGP: Introduction

- Created by Phil Zimmerman in 1991 originally using symmetric encryption
- PGP 3 allowed for asymmetric encryption
- Zimmerman's team and Viacrypt (who'd licensed RSA from RSADSI) merged to form PGP Inc in 1996
- OpenPGP as a standard proposed to IETF in 1997 to avoid patent issues.
- PGP Inc now owned by Symantec
- GPG is the Free Software Foundation's implementation of the OpenPGP standard

Asymmetric encryption refresher:

- One key mathematically related to the other.
- Public key can be generated from private key. But NOT vice versa.
- If you **encrypt** data with the **public** key, you need to **private** key to **decrypt**

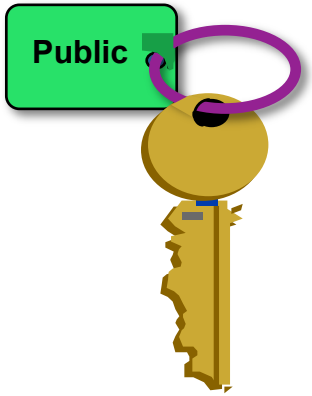
- You can **sign** data with the **private key** and **verify** the signature using the **public key**



keys

- Private key is kept SECRET.
- You **should** encrypt your private key with a **symmetric** passphrase.

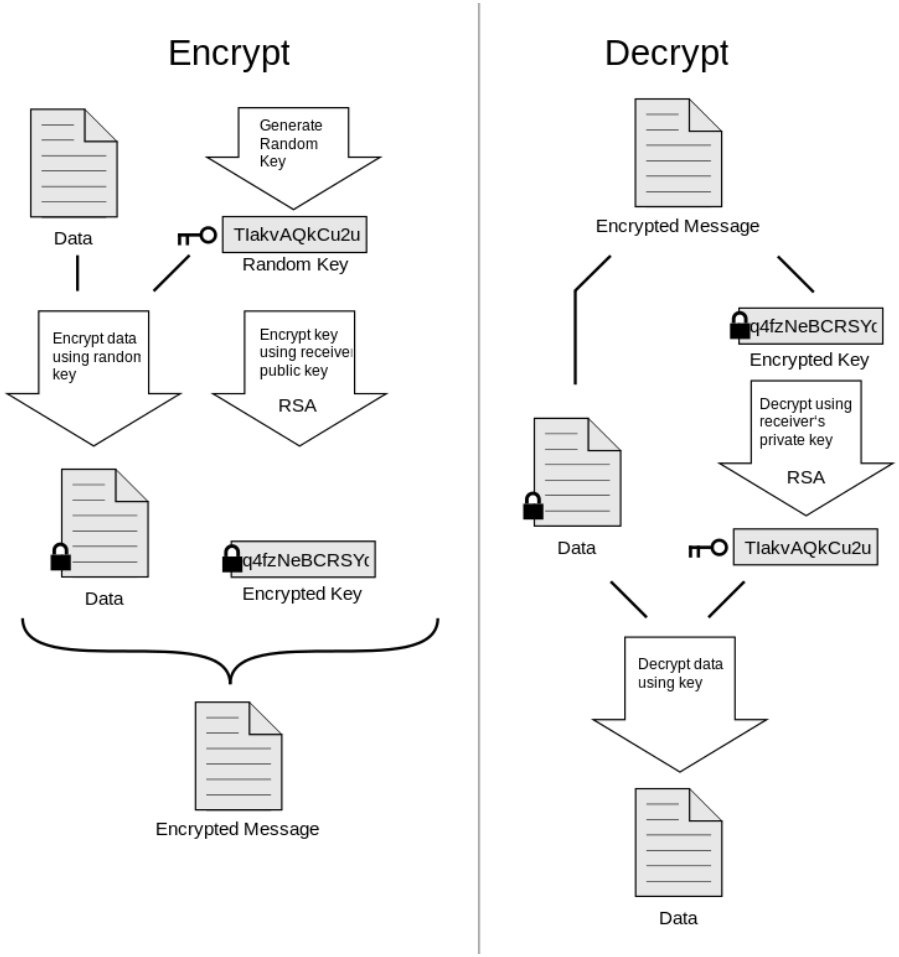
- Public key is distributed.
- Anyone who needs to send you confidential data can use your public key



Signing & Encrypting

- Data is encrypted with a public key to be decrypted with the corresponding private key.
- Data can be signed with the private key to be verified by anyone who has the corresponding public key.
- Since public keys are data, they can be signed too.
- Hash functions that generate fixed length fingerprints of any input data can be used to identify keys that would otherwise be over 1024 bits long

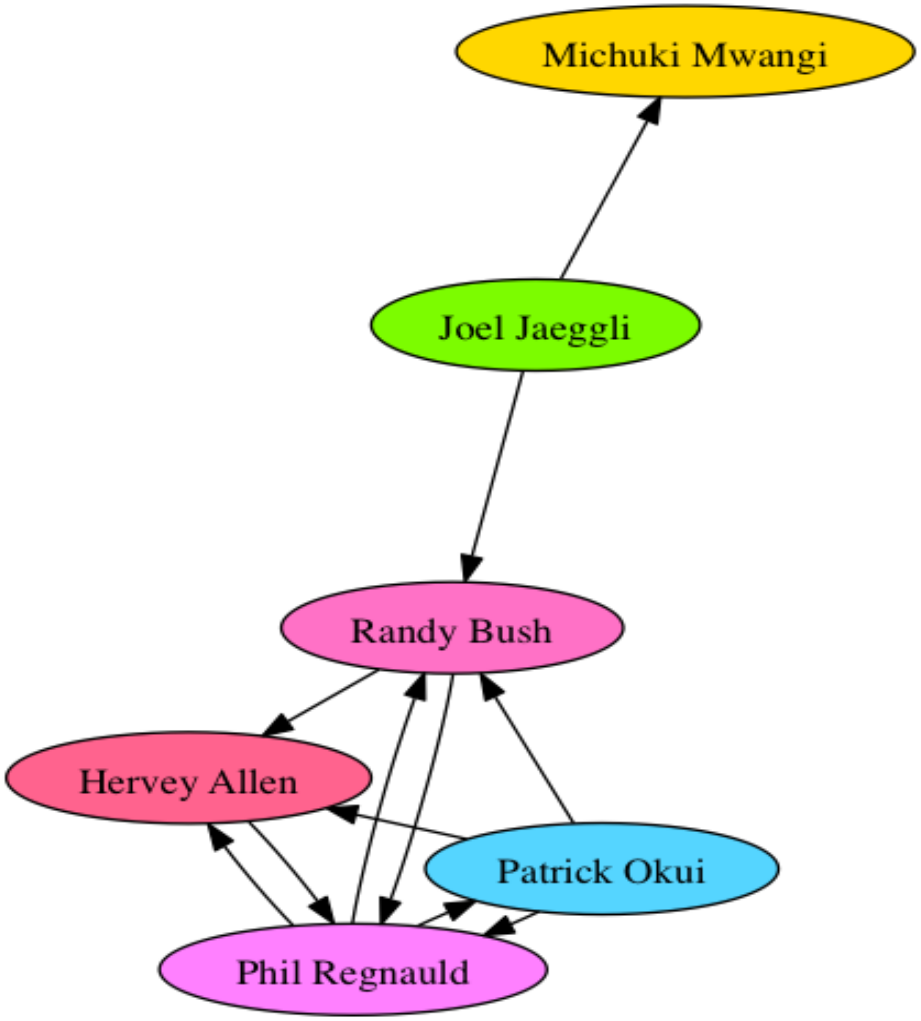
How PGP Works



Trust

- Centralized / hierarchal trust – where certain globally trusted bodies sign keys for every one else.
- Decentralized webs of trust – where you pick who you trust yourself, and decide if you trust who those people trust in turn.
- Which works better for what reasons?

Sample web of trust



Installing GnuPG Software

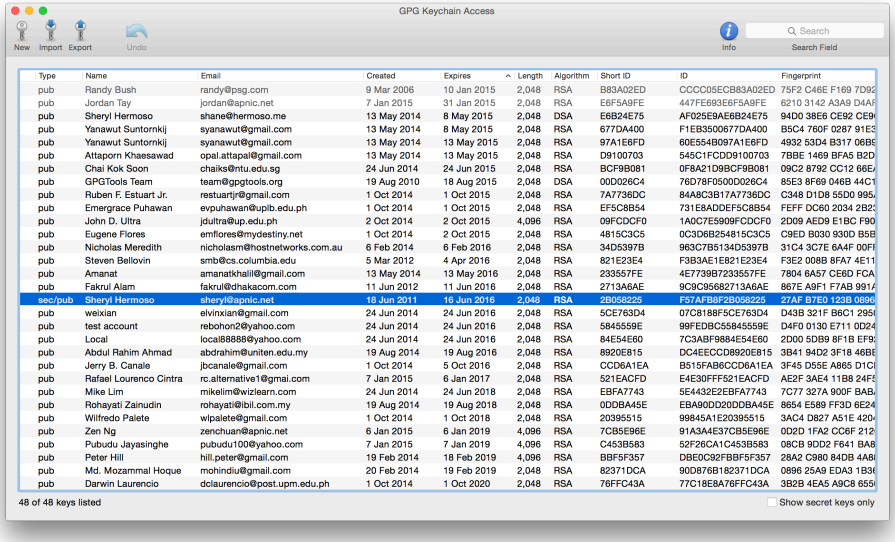
- GNU Privacy Guard
- Core software either commercial from pgp or opensource from gnupg
  - GPG4Win for windows
  - GPGTools for Mac OS X
- Your package manager for Linux/UNIX
  - Source code from https://www.gnupg.org/
- Written by Werner Koch in 1997
  - After attending a talk by Stallman

Key management: generation

- Using graphical tools based on what you installed above:
  - GPG Keychain Access for OS X
  - Kleopatra or GPA for windows
- Using the command line:
  - `gpg --gen-key`
- Generate a key – use your email address. The comment field can be left blank.

Key Management

- Using graphical tools, you may see your own keypair, and all imported public keys from contacts
  - Keychain Access (Mac OS X)
  - Kleopatra (Windows)



Key management - Distribution

- On printed media: published book or business cards:
- Digitally in email or using sneaker-net
- Online using the openpgp key servers.
- Still does not tell you if you trust the key.

Key management - Rollover

- Expiry dates ensure that if your private key is compromised they can only be used till they expire.
- Can be changed after creating the key.
- Before expiry, you need to create a new key, sign it with the old one, send the signed new one to everyone in your web of trust asking them to sign your new key.
- Many people create keys that don't expire. Think about the security implications of that.

Key management - Revocation

- Used to mark a key as invalid before its expiry date
- Always generate a revocation certificate as soon as you create your key
- Do not keep your revocation certificate with your private key
  - `gpg --gen-revoke IDENTITY`

Key management - Partying

- Key signing parties are ways to build webs of trust.
- Each participant carries identification, as well as a copy of their key fingerprint. (maybe some $ as well ☺)
- Each participant decides if they're going to sign another key based on their personal policy.
- Keys are easiest kept in a keyring on an openpgp keyserver in the aftermath of the party.

Interesting gpg commands

- Get help for gpg options
  - `gpg --help` AND `man gpg`
- Print the fingerprint of a particular key
  - `gpg --fingerprint IDENTITY`
- Export a public key to an ASCII armored file.
  - `gpg -a --output my-public-key.asc --export IDENTIY`

*IDENTITY = email or PGP key ID

Interesting gpg commands

- Import a key from a file into your keyring
  - `gpg --import  public.asc`
- Import a key from a keyserver
  - `gpg --recv-keys --keyserver hkp://keys.gnupg.net`
- Send your key to a keyserver
  - `gpg --send-keys --keyserver hkp://keys.gnupg.net`
- Sign a key
  - `gpg --sign-key IDENTITY`

Questions