Cryptographic Applications: Pretty Good Privacy

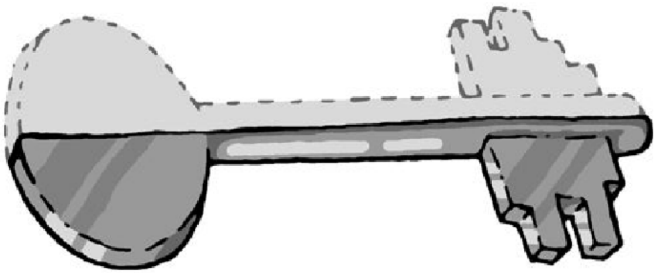
Acknowledgement

- Original slides made by
  - Patrik Okui <polui.psg.com>
  - Yoshinobu ‘maz’ Matsuzaki <maz@iij.ad.jp>

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**

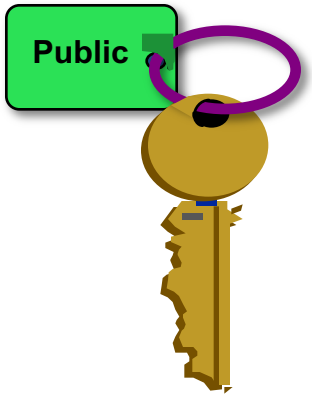
- You can **sign** data with the **private** **key** and **verify** the signature using the **public** **key**



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.

Use cases: email

- Enctypting: to send confidential information
  - Encrypt with a recipient's public key, and decrypt by using the recipient's private key
- Signing: to prove the message actually comes from signer and is not modified during delibery
  - Sign with signer's private key, and verify by using signer's public key

Use case: file distribution

- Signing: to prove that the contents is actually distributed by the signer and not modified since signed
  - Sign with signer's private key, and verify by using signer's public key
- You can generate a separate signature file if needed
  - You have the original file and corresponding signature file for it

Installing GnuPG Software

- Core software either commercial from pgp or opensource from gnupg.
- https://www.gpg4win.org/ for Windows
- https://www.gpgtools.org/ for OS X
  - This is now a commercial with 30 trial days
- Your package manager for Linux/UNIX
- Source code from https://www.gnupg.org/
- we have hands-on later

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.

Public key in armor format

```
$ gpg --export --armor keiichi@iij.ad.jp
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBFQswzIBEADDzpxoRx8V3pkX3XeMH7HetaJcqp+2ESNWTqrzYEv5wcpaJj1B
4Tau7G9kFDb/ieFYB1ASXm+h+kzo8cPzn8+nVQc9c2tqh5eYX01A1Xvm2QnIEfdU
2ce7PNSXgFtHILdyHnrtLnkTgPq+8qJkoK6f50hq3Vu7Uekz/MuHSFdgqIdNvVZo
tVFuxfouxHhMDfTciu4DddvftHvTF06FzajQdnSuBw6rZcA52rvT2ZZaEvV8rEsC
dwSk9h1DKwz/39PT2rlwfZE5KCBvu3AZgFlkeBusYLX41TgH1pi5s1RLUSU+jOZr

   (snip)

9P3uKnnfx0BkUE09oyRAfrTIBqXfrZeJy3UGuXe+UnM/udyRLlHDPEewpcsJtfyc
urxTU8UYno6BNeruXqW7dvbQGaAJS16MiCk6xYbLSsP6AjBEmdQwsO7FGgwwnC5r
yZlBIuKWt6BJNkjHl554UXIFAM2Qifnf5VC5h8Hgnp+x+JGeU7fR2W0a2ykTqoCl
kyx2MvuRb/1C77m/SYAR7h4ZFCFnXEuos5qosFJw2gx2MFUXCeOa0yQRzcWIRUcY
ir1pw2jIQl1gWngS7pwurEMhq5swUz1nOEdzyEtP6F8=
=3jxT
-----END PGP PUBLIC KEY BLOCK-----
```

Key management: distribution

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

Get the right key

- You don’t need to trust the person
- But you need to be confident that the key is owned by the person
- Check owner’s identity and integrity of the public key before use
  - Ask passport or appropriate ID card for identity check
    - Name is usually included in the public key
  - Ask fingerprint of the key to confirm that the public key you have is the same key which the person distributed

Fingerprint

```
$ gpg --fingerprint keiichi@iij.ad.jp
pub   rsa4096 2014-10-02 [SC] [expires: 2021-03-05]
      1B15 ACBC 1FB8 7325 B9BF  DCE6 B74D 1CFB 5173 3681
uid           [ultimate] Keiichi SHIMA (Internet Initiative Japan, Inc.) <keiichi@iij.ad.jp>
uid           [ultimate] Keiichi SHIMA (IIJ Innovation Institute, Inc.) <keiichi@iijlab.net>
uid           [ultimate] [jpeg image of size 3011]
uid           [ultimate] [jpeg image of size 14554]
sub   rsa4096 2014-10-02 [E] [expires: 2021-03-05]
sub   rsa4096 2015-02-12 [S] [expires: 2021-03-05]
```

Belonging

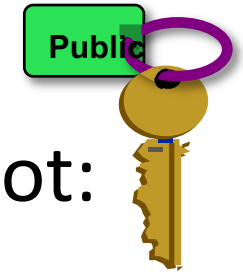
Passport

JAPAN

finger-print: 8E 9C EC 04 87 6B ... 46 3B CB F7 72 CE

Finger-print should match

Name should match



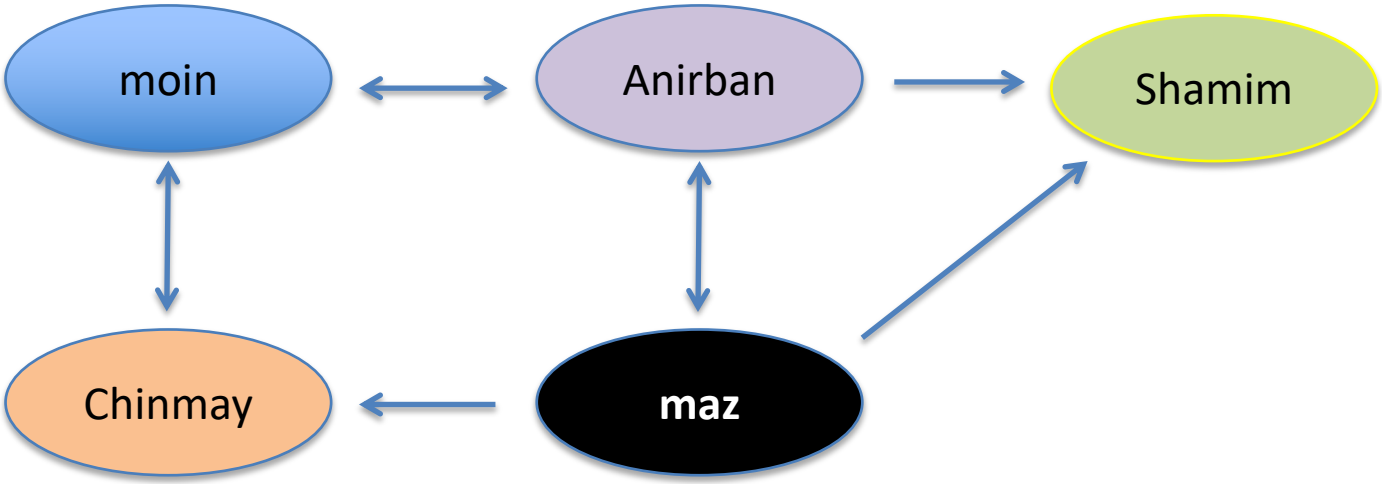
You got:

finger-print: 8E 9C EC 04 87 6B ... 46 3B CB F7 72 CE
uid: Matsuzaki Yoshinobu <maz@iij.ad.jp>

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.



You can share your “trust information” by publishing others’ public keys with your pgp sign

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.

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
- 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

Information in your pubkey

- Key and its algorithm
  - Creation and Expiration date
  - Preferences
- User ID (string)
  - Usually RFC822 style mail name like
  - Name Foo <foo@example.com>
- Subkeys

Subkey

- To have separate keys on purpose
  - Primary key to certificate other subkeys
- Purpose of keys
  - Certificate
  - Signing
  - Encryption
  - Authentication

Interesting gpg commands

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

Interesting gpg commands

- Import a key from a file into your keyring
  - `gpg --import public.asc`
- Locate a key by email address
  - `gpg --auto-key-locate keyserver --locate-keys someonesaddress@example.net`
- Import a key from a keyserver
  - `gpg --recv-keys KEYID`
- Send your key to a keyserver
  - `gpg --export youraddress@example.org | curl -T - https://keys.openpgp.org`
- Sign a key
  - `gpg --sign-key KEYID`

Interesting gpg commands

- Signing a file
  - `gpg --clearsign FILE`
  - `gpg --sign FILE`
- Verifying/Extracting a file
  - `gpg --verify FILE.gpg`
  - `gpg --decrypt FILE.gpg`
- Encrypting a file for a recipient
  - `gpg --recipient RECIPIENT_IDENTITY --encrypt FILE`

SKS Keyserver Attack

- In late June 2019, a major OpenPGP key server network was attacked
- A good summary can be found at
  - https://gist.github.com/rjhansen/67ab921ffb4084c865b3618d6955275f
- Some of the PGP software stopped using the existing keyservers and shifting to keys.openpgp.org
- The community is still looking for better key distribution mechanism, so stay tuned