Cryptographic Applications: Pretty Good Privacy

maz@iij.ad.jp

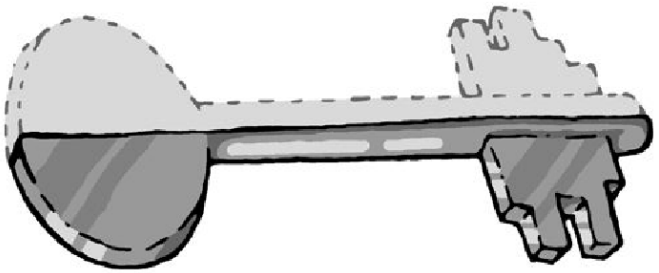
stole some slides from

pokui@nsrc.org

Asymmetric encryption

- 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**

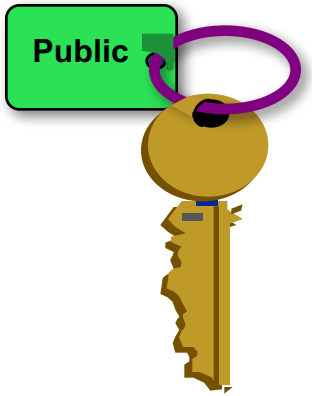
- You can **sign** data with the **private** key and **verify** the signature using the **public** key



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 case: email

- Encrypting: to send confidential information
  - Encrypting with a recipient's public key, and it's decrypt-able by using the recipient's private key
- Signing: to prove the message actually comes from signer and is not modified during delivery
  - Signing signer's private key, and it's verifiable 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
  - Signing by signer's private key, and it's verifiable by using signer's public key
- You can generate a separate signature file if needed
  - You have the original file and a corresponding signature file for it

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

-----BEGIN PGP PUBLIC KEY BLOCK-----

mQCNAzfU/toAAAEEAK22wkJ6+Nht4OkYw62AZPM3Kn0xGI8U0uossRyYdDWiP+6F
eEluQIDEfGa4gOOF1qO6rk0j5QyX41pxWOJ7MdnAlWfildcDPdTrGl1/q4aaflSa
RyLrOHz9xqV+xaPWlxadAd1Phh8ZREZbNAKtW7acIBthIL82ajoo7EGKtqONAAUT
tCNNYXRzdXpha2kgWW9zaGlub2J1IDxtYXpAaWlqLmFkLmpwPokAlQMFEDfU/to6
KOxBirajjQEB/YUEAKgqAjb27cikgKtBNCK29LtxhgJJzeTTtNZn9veQPMvPlMhx
xw5r3m9nwY8MGbCV+Z8OD+cEryPRhpB/wddtTMeeGZMLwO5m1jMLp0WVqSWT+c5m
6jmE06ZACED7dOSvpG0A1S33oP6kuad/1QQXIiOmJvzgRsoKWc1CfLyFodH0iEYE

    <snip>

2Tl1RRthvhBXSo23cW8An3oLfa/H2hw5nF5KdYLrKyLvJiX3iEYEExECAAYFAkF3
OEQACgkQQKY/75CQ6v4ofACfY2ETxrqc7xusWsH50F/M4gxsjCoAn39+EMB+1iPo
gOg+O7NZf1j6pKqWiEYEExECAAYFAkF3OFQACgkQIHzJaMhcykY/qQCfeAMUC/KY
PG3+8xeLxsio1muPxqkAoLvywwZVhVD5JRmX1TD3JeGYDnBGiEYEEhECAAYFAkGM
QU4ACgkQEC8OJ8ylIRogowCcCKi8vZvwh/vnR07EazW77RrhVZMAnRJ74vNcXjLg
mEQSksbAEAZG3N9D
=tqGr
-----END PGP PUBLIC KEY BLOCK-----

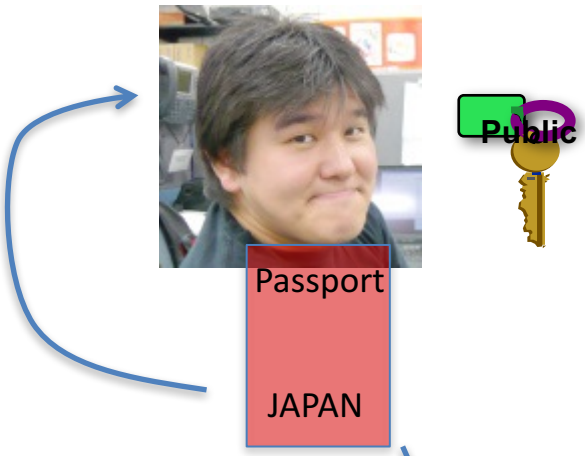
Key management: distribution

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

Get the right key

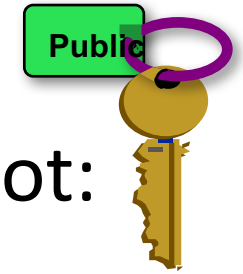
- Check owner's identity and integrity of the public key before use
  - Ask passport or appropriate ID card for identity check
    - Name is usually included in the public key
  - Ask fingerprint of the key to confirm that the public key you have is the same key which the person distributed

Checking



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

You got:

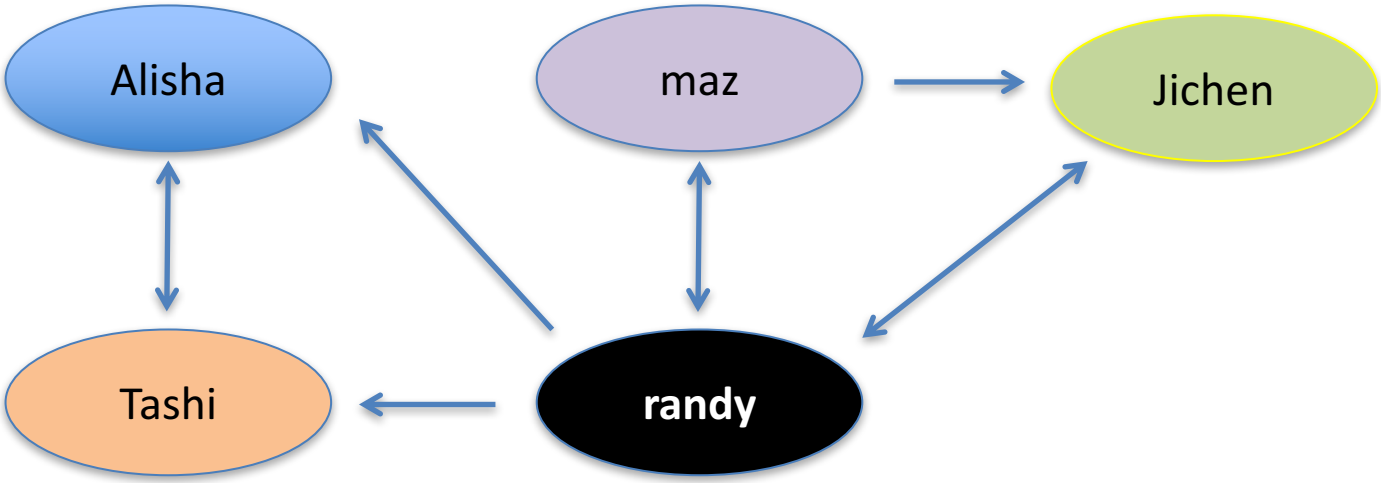


finger-print: 8E 9C EC 04 87 6B ... 46 3B CB F7 72 CE
uid: Matsuzaki Yoshinobu <maz@iij.ad.jp>

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 pair, sign your new public key by using the old key, distribute the signed new public key to everyone in your web of trust
  - You might ask 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 in the aftermath of the party.

Many keys…

Key management: use

- Need to specify a key to be used
  - keyid: hash value of the key
  - uid: name or email address in the key
- Email clients usually use ‘email address’ to specify a key
  - Be careful while you generate a new key pair, you should use your email address

Implementations

- GnuPG
  - https://gnupg.org/
- Gpg4win (for Windows)
  - https://www.gpg4win.org/
- GPG Suite (For Mac)
  - https://gpgtools.org/

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 -a --output my-public-key.asc --export IDENTIY`

Interesting gpg commands

- Import a key from a file into your keyring
  - `gpg --import FILE`
- 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`

Interesting gpg commands

- Signing a file
  - gpg --clearsign FILE
  - gpg --sign FILE
- Encrypting a file for a recepient
  - gpg -e FILE -r IDENTITY

clear signed mail

From: Matsuzaki Yoshinobu <maz@iij.ad.jp>
To: maz@iij.ad.jp
Content-Type: Text/Plain; charset=us-ascii

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

test
- -----
Matsuzaki Yoshinobu <maz@iij.ad.jp>
 - IIJ/AS2497 INOC-DBA: 2497*629
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1

iEYEARECAAYFAlX2kAAACgkQf+1KnE2/CBdemQCeLj6o9h8GGH9XjFYA22SqhvMN
5mIAnRcf2iqDti+FJX8sQWmOd+/+dP3w
=TVB8
-----END PGP SIGNATURE-----

clear encrypted mail

From: Matsuzaki Yoshinobu <maz@iij.ad.jp>
To: maz@iij.ad.jp
Content-Type: Text/Plain; charset=us-ascii

-----BEGIN PGP MESSAGE-----
Version: GnuPG v1

hQEOA6AuGqmvovGNEAP+LATBSvJo7VswYkLj1D83m0KtNACcSZdkRdWcJXQEtYTr
/+by0t1u+UFQlBtmyLxX8PauYTuKqa1UFtSkLcMAUmWamgzT6ArS18y43EXGBR7z
6Dz2gIp0VEvZeZ4kZO06NOtewyT3XdvL1D0/rRJSZsK4vaFlR18+EBR5x60VmzsD
/jw/iBtWcTl7PAjaq7WpunqX5vUIQGASlGwQBTaReHUnqIPJZ/GdNwDoW8Bxd83O
8IBWQJ9Oq7bjx/BdTlp90HHQISq8D0ryQSP+chqBFNpHLwu9+jzKFVJ1hWoQ6Vrh
rszEIVyChv4VPZDK+YnHfQfU0fPy3RQ3k5CmUD7Dg+gS0pIBbZOVfgXjwU8uHNCL
3Gv4X+aUYBJa5dYPfh35ksKucbaXFR5diXGqYH0x0ByKqaU66l0BFZ3F8uqcHesH
PbMc0R9E/9hBtrGf3oLjdarb3TJ3+Z0585NedzCA2ifEcH/k30NB14gAlNJMYK2B
6hW+NBxtcX3qVch3tGmBEtV6B267HtLOe16NUWiAsPY9lk4K9w==
=UUSj
-----END PGP MESSAGE-----

S/MIME signed mail

```
From: Matsuzaki Yoshinobu <maz@iij.ad.jp>
To: maz@iij.ad.jp
Content-Type: Multipart/Signed; protocol="application/pgp-signature";
 micalg=pgp-sha1;
 boundary="--Security_Multipart(Mon_Sep_14_18_21_21_2015_551)--”

----Security_Multipart(Mon_Sep_14_18_21_21_2015_551)--
Content-Type: Text/Plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

test
-----
Matsuzaki Yoshinobu <maz@iij.ad.jp>
 - IIJ/AS2497  INOC-DBA: 2497*629

----Security_Multipart(Mon_Sep_14_18_21_21_2015_551)--
Content-Type: application/pgp-signature
Content-Transfer-Encoding: 7bit

-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1

iEYEABECAAYFAlX2kZEACgkQf+1KnE2/CBf5UQCdGQEVWaxIIYqrT1JgUiy0jrT3
GI4An2japtU32kMkC+1pSV7wjEFI7INI
=PFwv
-----END PGP SIGNATURE-----

----Security_Multipart(Mon_Sep_14_18_21_21_2015_551)----
```

S/MIME encrypted mail

From: Matsuzaki Yoshinobu <maz@iij.ad.jp>
To: maz@iij.ad.jp
Content-Type: Multipart/Encrypted; protocol="application/pgp-encrypted";
 boundary="--Security_Multipart(Mon_Sep_14_18_23_10_2015_919)--”

----Security_Multipart(Mon_Sep_14_18_23_10_2015_919)--
Content-Type: application/pgp-encrypted
Content-Transfer-Encoding: 7bit

Version: 1

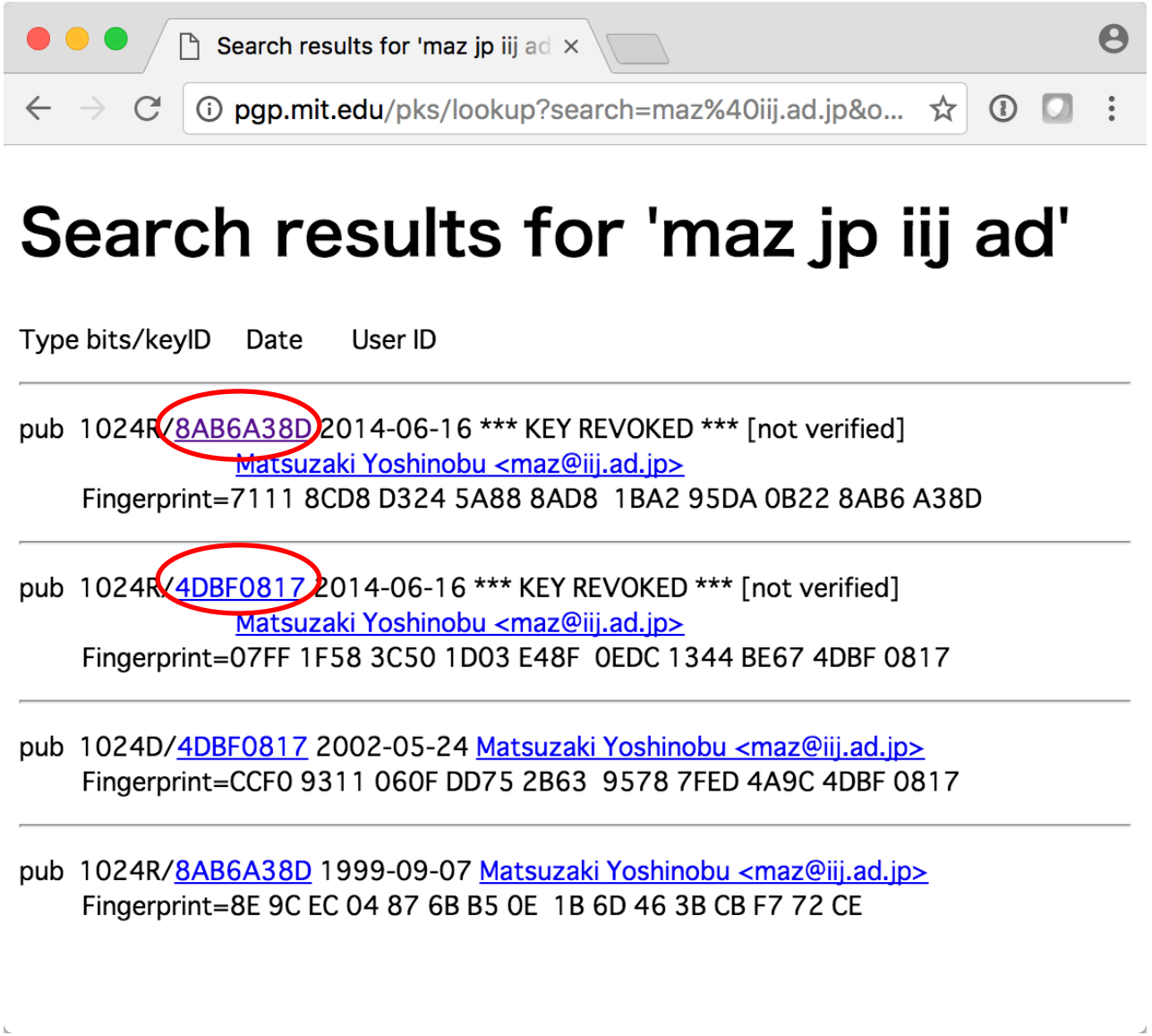
----Security_Multipart(Mon_Sep_14_18_23_10_2015_919)--
Content-Type: Application/Octet-Stream
Content-Transfer-Encoding: 7bit

-----BEGIN PGP MESSAGE-----
Version: GnuPG v1

hQEOA6AuGqmvovGNEAP/coU5eZDmt29PRSpYOKsP40HHHENRtJBRo9N2u5QlfkEM
fd6PGhIouxajTscvsFHOmZOj4Q9rQr+mcVivgGDJt33CdaioFcYNwf059ndMFX1N
hdRHVvTLN+ma9IrEWOJdoYKr1NMje7zvjbQulWcvs8Fx9wdOi7nj2MDVzzN3WwgE
AKFBzz9gNmiy2+DaOC6uLwBmiYe2YbdSZ4iiV8aHnoKXJME7Fahaj/EmA8PHWELv
A5a5zeV+3/WSkjXR2qNpsKMMsJB/S8ZawQ3oRoV2QgKZIOwWXqnMT0sFinFh2hS+
fsMsCTjy9vUNXBYiU3jb+iEHjHUS4e6uvOHlfoWx+JT20sAcARMpBFaXvi0XAS3Z
mig7zv4joH2qHyDFq7f1Th//1TkUht571FFIGGtT8ypVwpGqTQaMkbczXBBE8Z+E
reQ/4sP7gwuv2JNg9GFGg3gyIyekI1z+Nal6uY7N3cSVl9RlpYBWxnMUeyeK47v/
QH9XLYqvAsL29yFVIciIn4kNFZni2bgrdCstHtXSoOY638ryaR37zNhmp9L6Ugxd
tA6/ohDyxVksanVf/a4vlwrJtlcPL3n4glj+AhY677W4CqCLFoZzeAJSeK3MMFYZ
jMLQ433/RqWB/4NCR2cuTg==
=8Nlq
-----END PGP MESSAGE-----

----Security_Multipart(Mon_Sep_14_18_23_10_2015_919)----

Be careful about a key on keyserver



NOT mine

mine

32bit keyid collision

- Modern computer can generate a new keypair which has the same 32bit keyid
  - You can generate a fake key by setting the same uid (Name and email) from the original one
- There is a research about it
  - https://evil32.com/
  - They generated fake keys to proof the concept, and make it publically available on the page

:(

- Someone uploaded the test data (a bunch of fake public keys) to a real keyserver
  - Anyone can upload any public key to a keyserver
- Anybody can not delete keys on keyserver once it's uploaded
  - It's strict policy
- The research group revoked the keys to minimalize confusions
  - But still it's confusable :(

Hands on

- Use an integrated software to exchange PGP messages
  - Backend might be GnuPG
- Use GnuPG software separately
  - GnuPG to sign/encrypt/decrypt a message, and send and receive it using your email client by copy and paste or as an attachment file

Gmail extensions

- E2Email
  - https://github.com/e2email-org/e2email
  - To build the app, you will need git, python, jdk1.7 or above, and ant
- Mymail-crypt for Gmail
  - https://chrome.google.com/webstore/detail/mymail-crypt-for-gmail/jcaobjhdnlpmopmjhijplpjhlplfkhba
  - known Issues like signing, importing key and so on

Thunderbird extension

- Enigmail
  - https://www.enigmail.net/index.php/en/
  - Can install GnuPG during setup procedure
  - How to setup
    - https://enigmail.wiki/

Gpg4win

- For Windows
- https://www.gpg4win.org/doc/en/gpg4win-compendium.html
  - Start from 6 Installing Gpg4win

GPG Suite

- For Mac
- https://gpgtools.tenderapp.com/kb/how-to/first-steps-where-do-i-start-where-do-i-begin-setup-gpgtools-create-a-new-key-your-first-encrypted-mail

hands-on 1

- Generate your pgp keypair

hands-on 2

1. Send me your public key
2. Send me a pgp signed message