GNU Privacy Guard
Best Practices

Kyeongmin Kim

What is PGP?

Pretty Good Privacy (PGP)

An encryption program that provides **cryptographic privacy** and **authentication** for data communication

Used for **signing**, **encrypting**, and **decrypting** texts, e-mails, files, directories, and whole disk partitions

Originally developed in 1991 by **Phil Zimmermann**

History of PGP

Zimmermann founded **PGP Inc** in 1996

PGP Inc and its intellectual property were acquired by **Network Associates Inc** (NAI) in December 1997

In 2002, NAI discontinued development and sales of PGP, and sold the rights to a new company, **PGP Corporation**

PGP Corporation was acquired by **Symantec** in 2010, and by **Broadcom** in 2019

What is OpenPGP?

A **non-proprietary** protocol for authenticating or encrypting data with using public key cryptography

Defines standard formats for encrypted messages, signatures, private keys, and certificates for exchanging public keys

The open standards version of NAI's PGP encryption protocol

The **OpenPGP Working Group** was formed in 1997 and is seeking the qualification of OpenPGP as an Internet Standard as defined by the Internet Engineering Task Force (IETF)

What is GPG?

GNU Privacy Guard (GnuPG or GPG)

A complete and free implementation of the OpenPGP standard as defined by **RFC 4880**

A **free software** distributed under the terms of the GNU General Public License

A part of the GNU Project

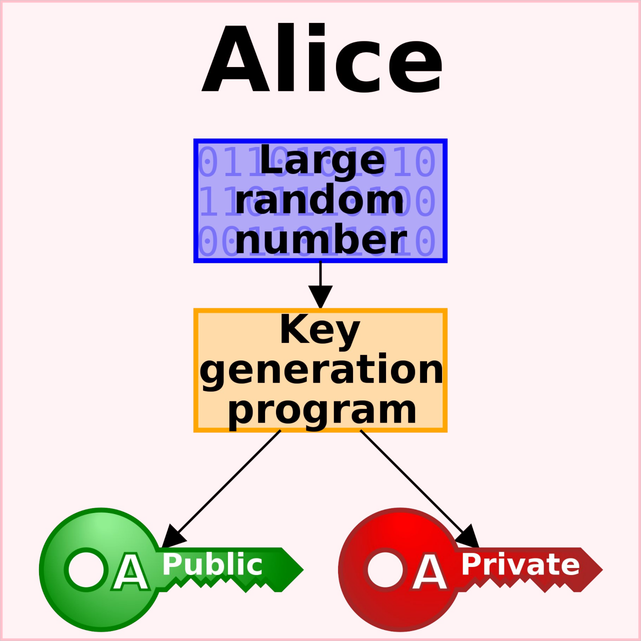
PGP vs OpenPGP vs GPG

PGP is the name of the original commercial software

OpenPGP is an open standard (RFC 4880) of PGP encryption software and the IETF standard compatible with the original PGP tool

GPG is a free software that implements the OpenPGP standard

Asymmetric Cryptography



Understanding the terminologies

A public key is revealed in the public.

A private key should be never disclosed to anyone.

A key pair consists one public key and one private key.

A key ring may have dozens of key pairs or just simply one.

The primary key indicates the default key pair of the key ring.

The sub keys on the other hand means the subsidiary ones from the key ring.

Key ID

Fingerprint: a **full** 40-character key identifier (e.g., 427F 11FD 0FAA 4B08 0123 F01C DDFA 1A3E 3687 9494)

Long ID: the last 16-characters of the fingerprint (e.g., DDFA 1A3E 3687 9494)

Short ID: the last 8 characters of the fingerprint (e.g., 3687 9494)

Someone can verify the authenticity of corresponding key with fingerprint

Always use the Fingerprint

Do not use or trust the abbreviated key ID because it can be forged by adversaries

Search Result of 0x00411886:

Fake Linus Torvalds: 0F6A 1465 32D8 69AE E438 F74B 6211 AA3B [0041 1886]

Real Linus Torvalds: ABAF 11C6 5A29 70B1 30AB E3C4 79BE 3E43 [0041 1886]

Search Result of 0x6092693E:

Fake Greg Kroah-Hartman: 497C 48CE 16B9 26E9 3F49 6301 2736 5DEA [6092 693E]

Real Greg Kroah-Hartman: 647F 2865 4894 E3BD 4571 99BE 38DB BDC8 [6092 693E]

How to obtain the genuine Fingerprint

Check the fingerprint on various websites (e.g., mailing lists, discussion forums, social media, personal websites). Check against PDFs, photographs, and videos in which the fingerprint appears (e.g., slides from a talk, on a T-shirt, or in the recording of a presentation). Ask people to post the fingerprint on various mailing lists, forums, and chat rooms. Repeat the above over Tor. Repeat the above over various VPNs and proxy servers. Repeat the above on different networks (work, school, internet cafe, etc.). Repeat the above from different computers and devices.

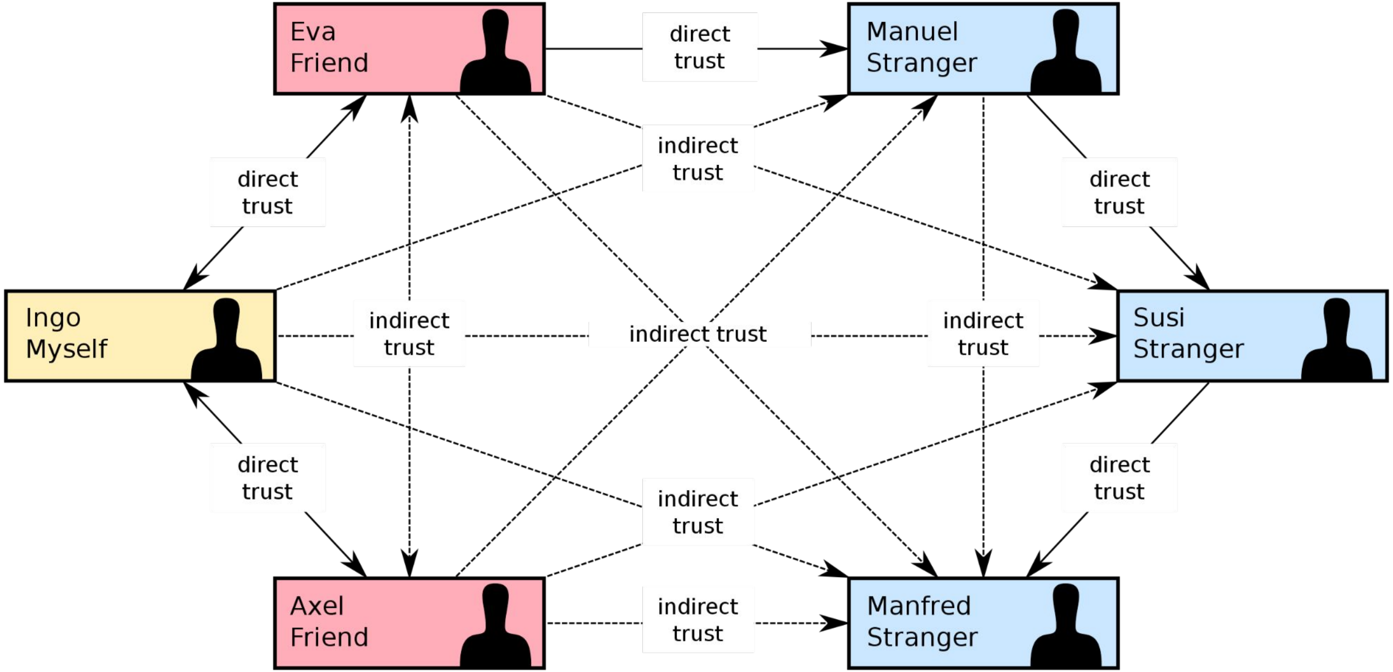
The best solution will be meeting up with people who know the fingerprint or using the Web of Trust.

Web of Trust

A concept used in PGP, GnuPG, and other OpenPGP-compatible systems to establish the authenticity of the binding between a public key and its owner.

Its decentralized trust model is an alternative to the centralized trust model of a public key infrastructure (PKI), which relies exclusively on a certificate authority (or a hierarchy of such).

Web of Trust


Eva
Friend
direct
trust
Manuel
Stranger
indirect
trust
direct
trust
direct
trust
Ingo
Myself
indirect
trust
indirect trust
indirect
trust
Susi
Stranger
direct
trust
direct
trust
indirect
trust
Axel
Friend
indirect
trust
Manfred
Stranger

Key Signing Party

An event at which **people present their public keys** to others in person, who, if they are confident **the key actually belongs to the person** who claims it, **digitally sign the certificate** containing that public key and the person's name, etc

A way to strengthen the web of trust as the OpenPGP public key infrastructure does not depend on a central key certifying authority, but to a distributed web of trust approach

Participants at a key signing party are expected to present adequate identity documents

Key Signing Party how-to

Key signing parties themselves generally do not involve computers, since that would give adversaries increased opportunities for subterfuge.

Rather, participants write down a string of letters and numbers, called a public key fingerprint, which represents their key.

The fingerprint is created by a cryptographic hash function, which condenses the public key down to a string which is shorter and more manageable.

Participants exchange these fingerprints as they verify each other's identification.

Then, after the party, they obtain the public keys corresponding to the fingerprints they received and digitally sign them.

Securing Private Keys

Storing your private keys in local disks may not a good idea.

Think about what happens if your system gets compromised by various reasons like malicious applications which try to get the /.gnupg directory.

Smart Card

V2.1 0005 00002E2A
0005
0000
2E2A
GnuPG
g10 code
Works with GnuPG
www.gnupg.org
BasicCard
ZeitControl
- Compatible to OpenPGP-Standard
- RSA with up to 4096 bit
- Key generation on card
- Hardware Random Number Generator
- Software download: www.gnupg.org

BANK CARD
Card
Maestro
1
2
3
4
5
6
7
8
9
0
C
X

USB Tokens emulating Smart Card

Nitrokey

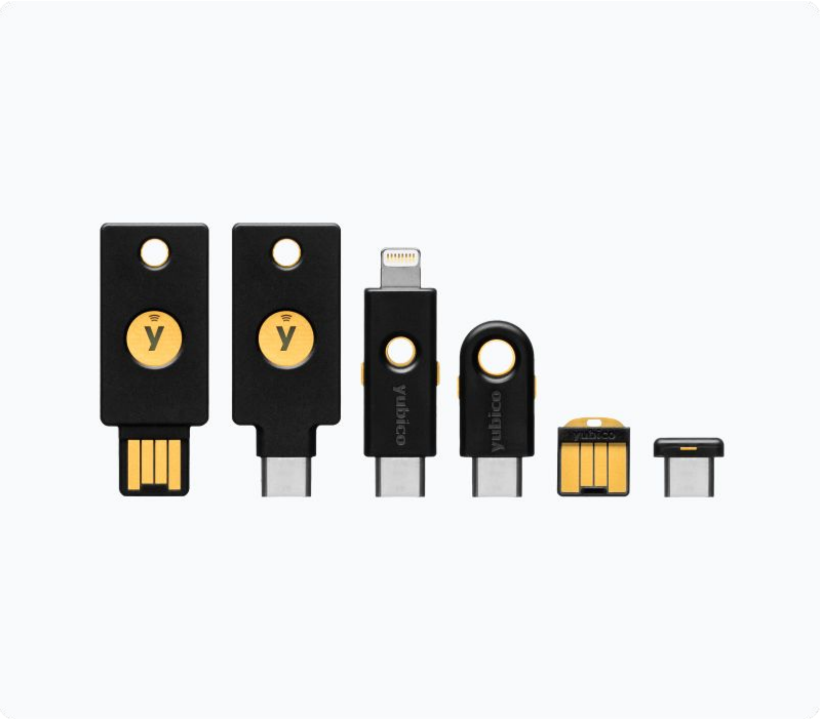
Used by Linux Kernel, Gentoo, and Arch developers.

An open source solution.

Yubikey

The largest USB token maker in the world.

But, its firmware is closed source.



Generating Keys

https://github.com/drduh/YubiKey-Guide

The guide has a tutorial based on Debian Live, but I recommend to use Tails.

Key Servers

https://keys.openpgp.org/

Use whatever key servers you would want to.

Do not use sks-keyservers.net since it is no longer maintained.

Searching by Short Key ID is not supported on keys.openpgp.org.

Protecting Email Communication

Thunderbird (https://www.thunderbird.net/en-US/) for Linux, Windows, macOS

K-9 Mail (https://k9mail.app/) with OpenKeychain (https://www.openkeychain.org/)

It was announced that K-9 Mail had been taken over by Mozilla in 2022 which distributes Thunderbird and will be rebranded to Thunderbird for Android.

Forward Secrecy

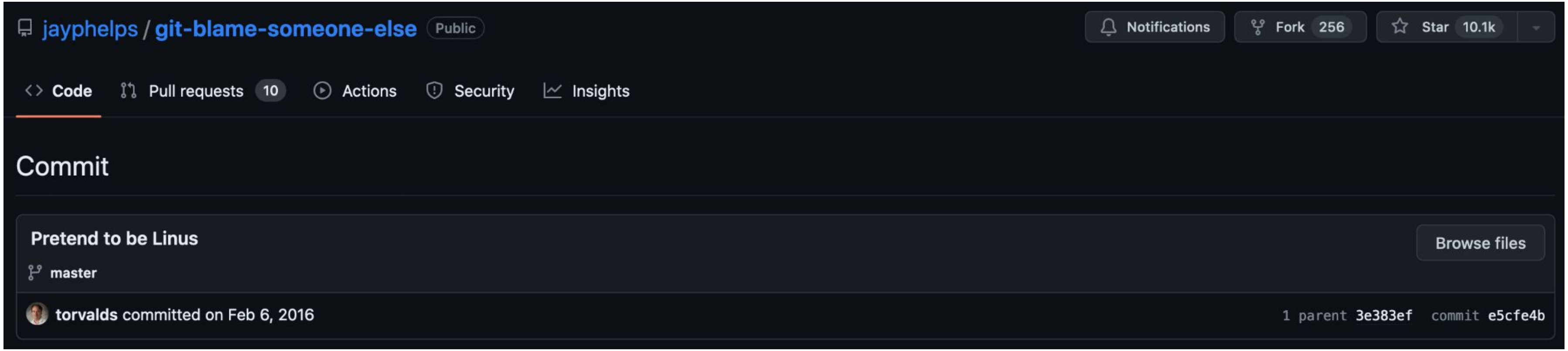
Forward Secrecy (FS), also known as Perfect Forward Secrecy (PFS)

A feature of specific key-agreement protocols that gives assurances that **session keys will not be compromised even if long-term secrets used in the session key exchange are compromised**

Protects past sessions against future compromises of keys or passwords

The OpenPGP standard does not support Perfect Forward Secrecy, so use a secure messenger like Signal which supports it if you need it.

Protecting the Code Integrity



Commit

Pretend to be Linus

master

torvalds committed on Feb 6, 2016

1 parent 3e383ef commit e5cfe4b

https://github.com/jayphelps/git-blame-someone-else/commit/e5cfe4bb2190a2ae406d5f0b8f49c32ac0f01cd7

GPG Integration in Git

```
$ git config --global user.signingKey <fingerprint>

$ git config --global commit.gpgSign true

$ git config --global tag.forceSignAnnotated true
```

git commit …

```
$ git commit -S -m "<commit_message>"

$ git verify-commit <object_hash>
```

git tag …

```
$ git tag -s <tag_name>

$ git tag -asm "<tag_message>" <tag_name>

$ git tag -s <tag_name> -m "<tag_message>"
```

Verifying Git Objects

```
$ git log --pretty=short --show-signature

$ git verify-tag <tag_name>

$ git merge --verify-signatures -S <merged-branch>
```

References

https://en.wikipedia.org/wiki/Pretty_Good_Privacy

https://www.openpgp.org/

https://en.wikipedia.org/wiki/GNU_Privacy_Guard

https://www.gnupg.org/

https://en.wikipedia.org/wiki/Public-key_cryptography

https://en.wikipedia.org/wiki/Web_of_trust

https://en.wikipedia.org/wiki/Key_signing_party

https://github.com/lfit/itpol/blob/master/protecting-code-integrity.md

https://www.qubes-os.org/security/verifying-signatures/

https://en.wikipedia.org/wiki/OpenPGP_card

https://en.wikipedia.org/wiki/Forward_secrecy

https://lkml.org/lkml/2016/8/15/445