GnuPG 实践
A Practical Guide to GnuPG

lch

lchopn@gmail.com

2017 年 11 月 19 日

Index

Index

GnuPG 简介

- GnuPG 是一个用于加密、数字签名及产生非对称匙对的软件。
- GnuPG 是自由软件基金会（Free Software Foundation）的 GNU 计划（GNU's not Unix）的一部分。

从 PGP（Pretty Good Privacy）说起

- 由 Phil R. Zimmermann 在 1991 年开发并免费于网络上发放。

从 PGP（Pretty Good Privacy）说起

- 由 Phil R. Zimmermann 在 1991 年开发并免费于网络上发放。
- 由于美国的出口限制，PGP 二进制程序、源代码在法律上不允许被传播到美国本土以外。

从 PGP（Pretty Good Privacy）说起

- 由 Phil R. Zimmermann 在 1991 年开发并免费于网络上发放。
- 由于美国的出口限制，PGP 二进制程序、源代码在法律上不允许被传播到美国本土以外。
- PGP 的作者将源代码由 MIT Press 出版并印刷成了一本书《PGP Source and Internals》，由于书籍的出口受到美国宪法第一修正案的保护，任何人都可以以 $60 购买这本书，并将源代码通过 OCR 扫描并录入电脑，并使用编译器编译为二进制文件。

从 PGP（Pretty Good Privacy）说起

- 由 Phil R. Zimmermann 在 1991 年开发并免费于网络上发放。
- 由于美国的出口限制，PGP 二进制程序、源代码在法律上不允许被传播到美国本土以外。
- PGP 的作者将源代码由 MIT Press 出版并印刷成了一本书《PGP Source and Internals》，由于书籍的出口受到美国宪法第一修正案的保护，任何人都可以以 $60 购买这本书，并将源代码通过 OCR 扫描并录入电脑，并使用编译器编译为二进制文件。
- PGP 仍然是商业软件，属于 PGP Inc.（Zimmermann 作为公司 CTO 与 Chairman），后并入 Network Associate （NAI，McAfee）现在已经被 Symantec 收购。

从 PGP（Pretty Good Privacy）说起

- 由 Phil R. Zimmermann 在 1991 年开发并免费于网络上发放。
- 由于美国的出口限制，PGP 二进制程序、源代码在法律上不允许被传播到美国本土以外。
- PGP 的作者将源代码由 MIT Press 出版并印刷成了一本书《PGP Source and Internals》，由于书籍的出口受到美国宪法第一修正案的保护，任何人都可以以 $60 购买这本书，并将源代码通过 OCR 扫描并录入电脑，并使用编译器编译为二进制文件。
- PGP 仍然是商业软件，属于 PGP Inc.（Zimmermann 作为公司 CTO 与 Chairman），后并入 Network Associate （NAI，McAfee）现在已经被 Symantec 收购。
- 在 1997 年 Zimmermann 与 IETF 共同制订 OpenPGP 标准（RFC1991 等）。

OpenPGP 与 GnuPG

- GnuPG （GNU Privacy Guard）基于 OpenPGP 标准开发，是 OpenPGP 标准的一个实现。
- GnuPG 最初由德国人 Werner Koch 开发。
- GnuPG 以 GNU Public License 发布，是自由软件。

Index

对称加密与非对称加密

对称加密

使用同一个密钥加密和解密（DES、AES）

非对称加密

使用不同的密钥加密解密（RSA、ECC）

Index

获取 GnuPG

GNU/Linux

大多数 GNU/Linux 发行版自带 GnuPG。

Windows

Gpg4win https://www.gpg4win.org/download.html

macOS

GPG Suite https://gpgtools.org/

Android

OpenKeychain https://www.openkeychain.org/

Index

Index

使用 GnuPG 2

- 请使用 GnuPG 2。

From manpage

```
This is the standalone verion of gpg. For desktop use you
should consider using gpg2 from GnuPG-2 Package.
... GnuPG 1.x, which is might be better suited for server and
embedded platforms...
```

使用 GnuPG 2

- 请使用 GnuPG 2。
- GnuPG 1 与 2 的区别

From manpage

```
This is the standalone verion of gpg. For desktop use you
should consider using gpg2 from GnuPG-2 Package.
... GnuPG 1.x, which is might be better suited for server and
embedded platforms...
```

使用 GnuPG 2

- 请使用 GnuPG 2。
- GnuPG 1 与 2 的区别
  - GnuPG 2 提供新的加密算法。

From manpage

```
This is the standalone verion of gpg. For desktop use you
should consider using gpg2 from GnuPG-2 Package.
... GnuPG 1.x, which is might be better suited for server and
embedded platforms...
```

使用 GnuPG 2

- 请使用 GnuPG 2。
- GnuPG 1 与 2 的区别
  - GnuPG 2 提供新的加密算法。
  - GnuPG 1 作为独立版拥有最小依赖，提供给嵌入式系统或服务器。

From manpage

```
This is the standalone verion of gpg. For desktop use you
should consider using gpg2 from GnuPG-2 Package.
... GnuPG 1.x, which is might be better suited for server and
embedded platforms...
```

使用 GnuPG 2

- 请使用 GnuPG 2。
- GnuPG 1 与 2 的区别
  - GnuPG 2 提供新的加密算法。
  - GnuPG 1 作为独立版拥有最小依赖，提供给嵌入式系统或服务器。
  - GnuPG 1 为了兼容可能还提供一部分对 PGP 2.6 的支持。

From manpage

```
This is the standalone verion of gpg. For desktop use you
should consider using gpg2 from GnuPG-2 Package.
... GnuPG 1.x, which is might be better suited for server and
embedded platforms...
```

不要使用简单生成

最新版本的 GnuPG 在使用 `gpg2 --gen-key` 命令的时候会默认产生 RSA-2048 bit 的一个主密钥和一个子密钥，并且主密钥默认开启签名和认证功能，子密钥默认开启加密功能，并且采用 SHA1 作为摘要算法。这一默认设置存在问题，故要尽量避免使用简单的 `--gen-key` 参数生成密钥。

编辑配置文件

在开始之前，你需要编辑 `$HOME/.gnupg/gpg.conf`、`$HOME/.gnupg/gpg-agent.conf`，以提高安全性！

生成密钥对

- 生成认证用主 Key
- 增加签名用子 Key
- 增加加密用子 Key
- 增加认证用子 Key

我该使用多少位的 RSA 密钥?

- 根据官方文档，2048 bit 的密钥理论上可以安全使用至 2020 年，并且在默认创建的密钥中使用了 2048bit 的密钥，也没有推荐或者反对使用更长的密钥。
- 尽管如此，我们仍然强烈建议在使用 RSA 密钥的时候选择更长的密钥，强烈推荐使用 4096bit 的密钥。

Index

加密 / 解密

加密

```
gpg2 --encrypt --sign --recipient john@example.com --armor
--output /tmp/very-secret-message.gpg /tmp/clear-text.txt
```

解密

```
gpg2 --decrypt --output /tmp/clear-text.txt
/tmp/very-secret-message.gpg
```

发布与签名

发布你的公钥

```
gpg2 --send-keys 0x09F33B41FEDB1EE9
```

签名一个 Key

验证签名信息!!!

- 签名方 `gpg2 --list-public-keys --with-icao-spelling 0x09F33B41FEDB1EE9`

发布与签名

发布你的公钥

```
gpg2 --send-keys 0x09F33B41FEDB1EE9
```

签名一个 Key

验证签名信息!!!

- 签名方 `gpg2 --list-public-keys --with-icao-spelling 0x09F33B41FEDB1EE9`
- 被签名方 `gpg2 --list-secret-keys --with-icao-spelling 0x09F33B41FEDB1EE9`

发布与签名

发布你的公钥

```
gpg2 --send-keys 0x09F33B41FEDB1EE9
```

签名一个 Key

验证签名信息!!!

- 签名方 `gpg2 --list-public-keys --with-icao-spelling 0x09F33B41FEDB1EE9`
- 被签名方 `gpg2 --list-secret-keys --with-icao-spelling 0x09F33B41FEDB1EE9`
- 签名 `gpg2 --sign-key john@example.com`

发布与签名

发布你的公钥

```
gpg2 --send-keys 0x09F33B41FEDB1EE9
```

签名一个 Key

验证签名信息!!!

- 签名方 `gpg2 --list-public-keys --with-icao-spelling 0x09F33B41FEDB1EE9`
- 被签名方 `gpg2 --list-secret-keys --with-icao-spelling 0x09F33B41FEDB1EE9`
- 签名 `gpg2 --sign-key john@example.com`
- 发送签名信息 `gpg2 --send-keys john@example.com`

发布与签名

发布你的公钥

```
gpg2 --send-keys 0x09F33B41FEDB1EE9
```

签名一个 Key

验证签名信息!!!

- 签名方 `gpg2 --list-public-keys --with-icao-spelling 0x09F33B41FEDB1EE9`
- 被签名方 `gpg2 --list-secret-keys --with-icao-spelling 0x09F33B41FEDB1EE9`
- 签名 `gpg2 --sign-key john@example.com`
- 发送签名信息 `gpg2 --send-keys john@example.com`
- 验证用签名 `gpg2 --lsign-key john@example.com`

密钥管理

生成吊销证书

```
gpg2 --output gpg-0x09F33B41FEDB1EE9.asc --gen-revoke
0x09F33B41FEDB1EE9
```

务必保证这一吊销证书保存在一个**安全**的地方，这一文件将能**直接吊销**你的密钥。如果可能的话，将其手写在纸上（一些打印系统甚至会记录你打印的东西），并保存在一个私密的地方。

密钥存储

- 在线存储

密钥存储

- 在线存储
  - 主密钥保存在本地，要求你必须信任你在本地安装的所有软件。(非常不安全)

密钥存储

- 在线存储
  - 主密钥保存在本地，要求你必须信任你在本地安装的所有软件。(非常不安全)
- 离线存储

密钥存储

- 在线存储
  - 主密钥保存在本地，要求你必须信任你在本地安装的所有软件。(非常不安全)
- 离线存储
  - 智能卡（YubiKey、GnuK 等等)

密钥存储

- 在线存储
  - 主密钥保存在本地，要求你必须信任你在本地安装的所有软件。(非常不安全)
- 离线存储
  - 智能卡（YubiKey、GnuK 等等)
  - USB 闪存、Tails

密钥存储

- 在线存储
  - 主密钥保存在本地，要求你必须信任你在本地安装的所有软件。(非常不安全)
- 离线存储
  - 智能卡（YubiKey、GnuK 等等)
  - USB 闪存、Tails
  - 一台独立的、离线的电脑

删除主密钥的私钥

- 首先备份整个 `~/.gnupg` 文件夹到 U 盘上。

删除主密钥的私钥

- 首先备份整个 `~/.gnupg` 文件夹到 U 盘上。
- 如果你的 GnuPG 版本 $>$ 2.1，那么只需要删除
  `$HOME/.gnupg/private-keys-v1.d/KEYGRIP.key`

删除主密钥的私钥

- 首先备份整个 `~/.gnupg` 文件夹到 U 盘上。
- 如果你的 GnuPG 版本 $>$ 2.1，那么只需要删除 `$HOME/.gnupg/private-keys-v1.d/KEYGRIP.key`
- 如果你的 GnuPG 版本 $<$ 2.1

删除主密钥的私钥

- 首先备份整个 `~/.gnupg` 文件夹到 U 盘上。
- 如果你的 GnuPG 版本 $>$ 2.1，那么只需要删除
  `$HOME/.gnupg/private-keys-v1.d/KEYGRIP.key`
- 如果你的 GnuPG 版本 $<$ 2.1
  - 你需要导出所有子密钥
    ```
    gpg2 --output secret_subkeys --export-secret-subkeys
    YOURMASTERKEYID
    ```

删除主密钥的私钥

- 首先备份整个 `~/.gnupg` 文件夹到 U 盘上。
- 如果你的 GnuPG 版本 $>$ 2.1，那么只需要删除

  `$HOME/.gnupg/private-keys-v1.d/KEYGRIP.key`
- 如果你的 GnuPG 版本 $<$ 2.1
  - 你需要导出所有子密钥

    ```
    gpg2 --output secret_subkeys --export-secret-subkeys
    YOURMASTERKEYID
    ```
  - 删除主密钥私钥

    ```
    gpg --delete-secret-keys YOURMASTERKEYID
    ```

删除主密钥的私钥

- 首先备份整个 `~/.gnupg` 文件夹到 U 盘上。
- 如果你的 GnuPG 版本 $>$ 2.1，那么只需要删除

  `$HOME/.gnupg/private-keys-v1.d/KEYGRIP.key`
- 如果你的 GnuPG 版本 $<$ 2.1
  - 你需要导出所有子密钥

    ```
    gpg2 --output secret_subkeys --export-secret-subkeys
    YOURMASTERKEYID
    ```
  - 删除主密钥私钥

    ```
    gpg --delete-secret-keys YOURMASTERKEYID
    ```
  - 导入子密钥私钥 `gpg --import secret_subkeys`

删除主密钥的私钥

- 首先备份整个 `~/.gnupg` 文件夹到 U 盘上。
- 如果你的 GnuPG 版本 $>$ 2.1，那么只需要删除
  `$HOME/.gnupg/private-keys-v1.d/KEYGRIP.key`
- 如果你的 GnuPG 版本 $<$ 2.1
  - 你需要导出所有子密钥
    ```
    gpg2 --output secret_subkeys --export-secret-subkeys
    YOURMASTERKEYID
    ```
  - 删除主密钥私钥
    ```
    gpg --delete-secret-keys YOURMASTERKEYID
    ```
  - 导入子密钥私钥 `gpg --import secret_subkeys`
  - 删除导出文件 `secret_subkeys`

- 此时可以检查 `gpg2 --list-keys`，私钥前的 `sec` 已经变成 `sec#`。表明主密钥私钥并不真正保存在此处。
- 此时还可以使用 `gpg --edit-key YOURMASTERKEYID passwd` 修改密码，防止日常使用中密码泄漏影响主密钥的安全。

Index

总结

- 务必保证私钥安全
- 再小心都不为过
- Take responsibility to your action.

参考文献与延伸阅读

- https://phab.enlightenment.org/w/gnupg/
- Debian Wiki: Subkeys
- GnuPG manpage
- The GNU Privacy Handbook
- An Advanced Intro to GnuPG
- Public-key cryptography - Wikipedia
- Symmetric-key algorithm - Wikipedia
- Pretty Good Privacy - Wikipedia
- GNU Privacy Guard - Wikipedia
- Applied Cryptography: Protocols, Algorithms, and Source Code in C, Bruce Schneier, Wiley

Thank you for your attention!