

本文由 Alex Gao 给您带来，有任何问题或疑问请邮件至:[mail@alexgao.com](mailto:mail@alexgao.com)。欢迎访问 <http://www.alexgao.com> (关注开源，关注 linux)

---

## 一、介绍

我们都知道，互联网是不安全的，但其上所使用的大部分应用，如 Web、Email 等一般都只提供明文传输方式（用 https、smtps 等例外）。所以，当我们需要传输重要文件时，应该对当中的信息加密。非对称密码系统是其中一种常见的加密手段。而在中国基于 PGP 方式加密的中文介绍少之又少，所以萌生了写一个完整教程的想法，当然本文大部分资料都是我搜遍网络整理出来的，并不能保证百分之百的原创 😊

GnuPG 是一个用来进行非对称加密(PGP)的免费软件，简称 GPG（是不是有的童鞋已经被 PGP 和 GPG 给搞昏了？ 😊）。先说说什么是非对称加密。传统的加密手段往往是使用同一个密码进行加密和解密。例如你加密时用的密码是“abc”，则解密时也要使用“abc”才行。这样就存在一个问题，你不能够把一段加密信息发送给你的朋友。试想，如果采用这种加密方式把信息发送给你的朋友时，你的朋友必须要知道你的密码才能把你的信息解密出来。但你如何保证你的朋友是绝对可靠的呢？也就是说，如果你的朋友把你的密码告诉了别人，你的密码就不再安全了。

非对称加密采用的是另一种思想。它会给你产生两个密钥，一个称为“公钥”，另一个称为“私钥”。公钥是可以公开的，你尽管把它传给别人；私钥你一定要保管好不让其他任何人知道。当某人得到你的公钥后，他就可以给你发送加密信息了。具体来说，他把他要发给你的信息用你的公钥加密后发给你，加密的信息只能用你的私钥去解密。这样，因为世界上除了你以外没有别人知道你的私钥，所以即使别人看到发送给你的加密信息他也无法解密，甚至连发送者本人也不行。因为他不知道你的私钥。简单说来，就是用公钥去加密；用对应的私钥去解密。想给谁发送加密信息，首先要得到他的公钥。

支持非对称加密的软件有多种，最著名的可能是美国的 PGP 了，不过它是个商业软件，价格不便宜。对于加密软件，我反对使用破解软件，因为如果信息需要加密的话，肯定是非常重要的信息，破解软件无法保证加密的安全可靠。因此我建议使用免费开源的 GnuPG 软件进行信息的加密和解密。

## 二、使用：

### 1.生成密钥对

要使用 GnuPG 加密，首先需要创建密钥对，执行：

```
=====
```

```
# gpg --gen-key
```

gpg (GnuPG) 1.4.5; Copyright (C) 2006 Free Software Foundation, Inc.

This program comes with ABSOLUTELY NO WARRANTY.

This is free software, and you are welcome to redistribute it under certain conditions. See the file COPYING for details.

请选择您要使用的密钥种类：

(1) DSA 和 ElGamal (默认)

(2) DSA (仅用于签名)

(5) RSA (仅用于签名)

您的选择？ 1 ←只有 1 可以用于加密，其他种类只能用于签名

DSA 密钥对会有 1024 位。

ELG-E 密钥长度应在 1024 位与 4096 位之间。

您想要用多大的密钥尺寸？(2048) ←选择密码的位数，位数越大，越安全，但速度越慢

您所要求的密钥尺寸是 2048 位

请设定这把密钥的有效期限。

0 = 密钥永不过期

<n> = 密钥在 n 天后过期

<n>w = 密钥在 n 周后过期

<n>m = 密钥在 n 月后过期

<n>y = 密钥在 n 年后过期

密钥的有效期限是？(0) 0 ←根据实际情况选择密钥期限

密钥永远不会过期

以上正确吗？(y/n)y ←确认

您需要一个用户标识来辨识您的密钥；本软件会用真实姓名、注释和电子邮件地址组合成用户标识，如下所示：

“Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>”

真实姓名：Hyphen Wang ←请填入真实姓名，后面会用到

电子邮件地址：[gpgencrypt@linuxfly.org](mailto:gpgencrypt@linuxfly.org) ←邮件作为标记之一，不能重复

注释：Use for GPG Encrypt ←仅是注释而已

您选定了这个用户标识：

“Hyphen Wang (Use for GPG Encrypt) <gpgencrypt@linuxfly.org>”

更改姓名(N)、注释(C)、电子邮件地址(E)或确定(O)/退出(Q)？O ←输入“O”确认

您需要一个密码来保护您的私钥。 ←输入两次用于访问私钥的密码，紧记，不能公开或丢失

我们需要生成大量的随机字节。这个时候您可以多做些琐事(像是敲打键盘、移动鼠标、读写硬盘之类的)，这会让随机数字发生器有更好的机会获得足够的熵数。

+++++++.....+++++++..+++++++

随机字节不够多。请再做一些其他的琐事，以使操作系统能搜集到更多的熵数！

(还需要 274 字节) ←运行一些的程序，以便在内存中获得更多随机数

我们需要生成大量的随机字节。这个时候您可以多做些琐事(像是敲打键盘、移动鼠标、读写硬盘之类的)，这会让随机数字发生器有更好的机会获得足够的熵数。

+++++++.....+++++++..+++++++  
+++^^^

gpg: 密钥 **A3942296** 被标记为绝对信任 ←密钥 ID

公钥和私钥已经生成并经签名。

gpg: 正在检查信任度数据库

gpg: 需要 3 份勉强信任和 1 份完全信任，PGP 信任模型

gpg: 深度：0 有效性： 2 已签名： 0 信任度：0-，0q，0n，0m，0f，2u

pub 1024D/A3942296 2008-12-19

密钥指纹 = E95E 1F77 6C4E 33BD 740C 19AB EEF9 A67E A394 2296

uid Hyphen Wang (Use for GPG Encrypt) <gpgencrypt@linuxfly.org>

sub 2048g/911E677B 2008-12-19

---

## 2. 密钥的回收

当您的密钥对生成之后，您应该立即做一个公钥回收证书，如果您忘记了您的私钥的口令或者您的私钥丢失或者被盗窃，您可以发布这个证书来声明以前的公钥不再有效。生成回收证书的选项是“-gen-revoke”。

```
gpg -output revoke.asc -gen-revoke mykeyID
```

其中 mykey 参数是可以表示的密钥标识，产生的回收证书放在 revoke.asc 文件里，一旦回收证书被发放，以前的证书就不能再被其他用户访问，因此以前的公钥也就失效了。

PS:如果一旦决定撤销已经上传的公钥，就需要将该密钥的回收证书上传至密钥服务器完成回收工作。

```
gpg -keyserver Server Address -send-keys mykeyID
```

## 3. 密钥的上传

当上述工作完成以后，为了让尽可能多的人获取您的公钥，您可以将公钥邮寄出去，或者贴在自己的个人主页上，当然还有一种更好的方法就是上传到全球性的密钥服务器，其他用户可以通过您提供的公钥 ID 来搜索并获得您的公钥。

通过如下命令可以将你的 key 发布到服务器上：

```
gpg -keyserver Server Address -send-keys mykeyID
```

PS:当然您也可以定义默认的服务器 key server，一般安装好后的默认 key server 都是 subkeys.pgp.net。你也可以通过修改.gnupg/gpg.conf中的 keyserver 信息来改变你的 key server。

#### 4. 密钥的导出 / 导入

我们通常需要导出公钥和私钥保存起来，当然公钥是可以满世界的泼洒，但是私钥请务必保存好，否则你的密钥对将会永久性的失去威力。

- 公钥的导出：

```
gpg -o keyfilename --export mykeyID
```

如果没有 mykeyID 则是备份所有的公钥，-o 表示输出到文件 keyfilename 中，如果加上-a 的参数则输出文本格式( ASCII )的信息，否则输出的是二进制格式信息。

- 私钥的导出：

```
gpg -o keyfilename --export-secret-keys mykeyID
```

如果没有 mykeyID 则是备份所有的私钥，-o 表示输出到文件 keyfilename 中，如果加上-a 的参数则输出文本格式的信息，否则输出的是二进制格式信息。

- 密钥的导入：

```
gpg --import filename
```

PS:用户可以使用 gpg --list-keys 命令查看是否成功导入了密钥。

#### 5. 加密解密和数字签名

通过上述的密钥生成以及公钥分发后，加密和解密数据变得非常容易，用户可以通过使用该功能来达到安全地在网络上传输自己的隐密数据的目的。

如果用户 patterson 要给用户 liyang 发送一个加密文件，则他可以使用 liyang 的公钥加密这个文件，并且这个文件也只有 liyang 使用自己的密钥才可以解密查看。下面给出加解密的步骤：

- 用户 patterson 使用 liyang 的公钥加密文件 test，使用下面的指令：

```
# gpg -e test
```

```
You did not specify a user ID. (you may use "-r")
```

```
Enter the user ID. End with an empty line: liyang
```

```
Added 1024g/C50E455A 2006-01-02 "liyang (hello) < liyang@sina.com>"
```

这样，就可以将 gpg.conf 文件加密成 test.gpg，一般用户是无法阅读的

PS:当然你也可以直接指定使用哪个用户的公钥进行加密:

```
gpg -e -r liyang test (-r 表示指定用户)
```

- 还可以加上参数 `-a` 来输出 ASCII 编码的文件 `test.asc` (`test.gpg` 是二进制编码的，不可用文本读)

```
gpg -ea -r liyang test
```

- **用户 liyang 使用自己的私钥来解密该文件，如下所示：**

```
# gpg -d test.gpg
```

You need a passphrase to unlock the secret key for

user: "liyang (hello) <liyang@sina.com>"

1024-bit ELG-E key, ID C50E455A, created 2006-01-02 (main key ID 378D11AF)

GnuPG 提示用户，需要输入生成私钥使用的密码：

Enter passphrase:

gpg: encrypted with 1024-bit ELG-E key, ID C50E455A, created 2006-01-02

"liyang (hello) <[liyang@sina.com](mailto:liyang@sina.com)>"

PS:无论加密解密，都可以加上 `-o` 参数来指定加密和解密后的输出文件，例如

```
#gpg -o doc.gpg -er name doc
```

其中 `name` 是选择谁的公钥加密，即谁是文件的接收者。

`doc` 为要加密的文件，即原文件

`doc.gpg` 为命令执行后生成的加密的文件，这里要先指定好文件名

- **对文件进行签名**

### 1、数字签名

命令格式：

```
#gpg -o doc.sig -s doc
```

其中 `doc` 是原文件，`doc.sig` 包含了原文件和签名，是二进制的。这个命令会要求你输入你的私钥的密码句。

```
#gpg -o doc.sig -ser name doc
```

既签名又加密

## 2、文本签名

```
#gpg -o doc.sig --clearsign doc
```

这样产生的 doc.sig 同样包含原文件和签名，其中签名是文本的，而原文件不变。

## 3、分离式签名

```
#gpg -o doc.sig -ab doc
```

doc.sig 仅包括签名，分离式签名的意思是原文件和签名是分开的。

b 表示分离式签名 detach-sign

## 4、验证签名

```
#gpg --verify doc.sig [doc]
```

验证之前必须导入文件作者的公钥，对于分离式签名，最后还要加上原文件，即后面的 doc。

### ● 密钥签名和用户信任(进阶功能)

尽管在理论上讲，具备了公钥和私钥就可以实现安全的信息通讯，但是在实际应用中，还必须对公钥进行有效确认。因为，确实存在伪造公钥信息的可能。

由此，在 GPG 中引入了一个复杂的信任系统，以帮助我们区分哪些密钥是真的，哪些密钥是假的。这个信任系统是基于密钥的，主要包括密钥签名。

当收到熟人的公钥并且 GPG 告知不存在任何实体可信信息附加于这个公钥后，首要的事情就是对这个密钥进行“指纹采样” (fingerprint)。例如，我们对来自 mike 的公钥进行了导入操作，并且 GPG 告知我们不存在这个密钥的附加可信信息，这时候，我们首先要做的工作就是对这个新密钥进行“指纹采样”，相关命令及执行情况如下：

```
$ gpg --fingerprint mike@hairnet.orgpub 1024D/4F03BD39 2001-01-15 Mike  
Socks (I'm WIRED) Key fingerprint = B121 5431 8DE4 E3A8 4AA7 737D 20BE  
0DB8 4F03 BD39sub 1024g/FDBB477D 2001-01-15$
```

这样，就从密钥数据中生成了其指纹信息，并且应该是唯一的。然后，我们打电话给 mike，确认两件事情。首先，他是否发送给我们了密钥；其次，他的公钥的指纹信息是什么。如果 Mike 确认了这两件事情，我们就可以确信这个密钥是合法的。接下来，我们对密钥进行签名操作，以表示这个密钥来自 Mike 而且我们对密钥的信任，相关命令及执行情况如下：

```
$ gpg --sign-key mike@hairnet.orgpub 1024D/4F03BD39 created: 2001-01-15  
expires: neversub 1024g/FDBB477D created: 2001-01-15 expires: never(1)  
Mike Socks (I'm WIRED) pub 1024D/4F03BD39 created: 2001-01-15 expires:  
neverFingerprint = B121 5431 8DE4 E3A8 4AA7 737D 20BE 0DB8 4F03  
BD39Mike Socks (I'm WIRED) Are you really sure that you want to sign this  
keywith your key: Ima User (I'm just ME) Really sign? yYou need a passphrase to  
unlock the secret key foruser: Ima User (I'm just ME) 1024-bit DSA key, ID
```

```
D9BAC463, created 2001-01-03Enter passphrase:$
```

执行到此，使用我们的私匙完成了对 Mike 的公匙的签名操作，任何持有我们的公匙的人都可以查证签名确实属于我们自己。这个附加到 Mike 的公匙上的签名信息将随它环游 Internet 世界，我们使用个人信誉，也就是我们自己的私匙，保证了那个密匙确实属于 Mike。这是一个多么感人的充满诚信的故事啊 :-)  
现实世界的人们是否应该从这严格的技术标准中反思些什么呢？

还是回到这里。获取附加于一个公匙上的签名信息列表的命令是：

```
gpg --check-sigs mike@hairnet.org
```

签名列表越长，密匙的可信度越大。其实，正是签名系统本身提供了密匙查证功能。假设我们接收到一个签名为 Mike 的密匙，通过 Mike 的公匙，我们验证出签名确实属于 Mike，那么我们就信任了这个密匙。推而广之，我们就可以信任 Mike 签名的任何密匙。

为了更加稳妥，GPG 还引入了另一个附加功能：可信级别（trust level）。使用它，我们可以为我们拥有的任何密匙的所有者指定可信级别。例如，即使我们知道 Mike 的公匙是可信的，但是事实上我们不能信任 Mike 在对其他密匙签名时的判断；我们会想，Mike 也许只对少数密匙进行了签名，但却没有好好地检查一遍。

设置可信级别的命令及执行情况如下：

```
$ gpg --edit-key mike@hairnet.orgpub 1024D/4F03BD39 created: 2001-01-15
expires: never trust: -/fsub 1024g/FDBB477D created: 2001-01-15 expires:
never(1) Mike Socks (I'm WIRED) Command> trust 1 = Don't know 2 = I do NOT
trust 3 = I trust marginally 4 = I trust fully s = please show me more information
m = back to the main menuYour decision? 2Command> quit$
```

在命令编辑环境中执行 trust，然后选择级别 2（I do NOT trust），这样我们割断了任何信任链，使每个密匙都必须经过 Mike 的签名。

## 6. 删除密钥

从私钥钥匙环里删除密钥：

---

```
# gpg --delete-secret-keys hyphenwang@redflag-linux.com
gpg (GnuPG) 1.4.5; Copyright (C) 2006 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions. See the file COPYING for details.

sec 1024D/A3942296 2008-12-19 Hyphen Wang (Use for GPG Encrypt)
```

```
<gpgencrypt@linuxfly.org>
```

```
要从钥匙环里删除这把密钥吗？(y/N)y
```

```
这是一把私钥！——真的要删除吗？(y/N)y
```

必须先删除私钥，然后才能删除公钥。

从公钥钥匙环里删除密钥：

```
# gpg --delete-keys hyphenwang@redflag-linux.com
```

```
gpg (GnuPG) 1.4.5; Copyright (C) 2006 Free Software Foundation, Inc.
```

```
This program comes with ABSOLUTELY NO WARRANTY.
```

```
This is free software, and you are welcome to redistribute it  
under certain conditions. See the file COPYING for details.
```

```
sec 1024D/A3942296 2008-12-19 Hyphen Wang (Use for GPG Encrypt)
```

```
<gpgencrypt@linuxfly.org>
```

```
要从钥匙环里删除这把密钥吗？(y/N)y
```

### 三.对称加密:

当然 GPG 同样具备普通的对称加密功能，这时候就不需要密钥，直接用密码加密即可（注意，这里的密码不一定是你私钥的密码，您大可以随意设定）

```
gpg -o doc.gpg -c doc
```

---

### 四.GPG 常用参数:

语法：gpg [选项] [文件名]

签字、检查、加密或解密

默认的操作依输入数据而定

指令：

-s, --sign [文件名] 生成一份签字

--clearsign [文件名] 生成一份明文签字

-b, --detach-sign 生成一份分离的签字

-e, --encrypt 加密数据

- c, -symmetric 仅使用对称加密
- d, -decrypt 解密数据(默认)
- verify 验证签字
- list-keys 列出密钥
- list-sigs 列出密钥和签字
- check-sigs 列出并检查密钥签字
- fingerprint 列出密钥和指纹
- K, -list-secret-keys 列出私钥
- gen-key 生成一副新的密钥对
- delete-keys 从公钥钥匙环里删除密钥
- delete-secret-keys 从私钥钥匙环里删除密钥
- sign-key 为某把密钥添加签字
- lsign-key 为某把密钥添加本地签字
- edit-key 编辑某把密钥或为其添加签字
- gen-revoke 生成一份吊销证书
- export 导出密钥
- send-keys 把密钥导出到某个公钥服务器上
- recv-keys 从公钥服务器上导入密钥
- search-keys 在公钥服务器上搜寻密钥
- refresh-keys 从公钥服务器更新所有的本地密钥
- import 导入/合并密钥
- card-status 打印卡状态
- card-edit 更改卡上的数据
- change-pin 更改卡的 PIN
- update-trustdb 更新信任度数据库
- print-md 算法 [文件] 使用指定的散列算法打印报文散列值

选项 :

- a, -armor 输出经 ASCII 封装
- r, -recipient 某甲 为收件者“某甲”加密
- u, -local-user 使用这个用户标识来签字或解密
- z N 设定压缩等级为 N (0 表示不压缩)
- textmode 使用标准的文本模式
- o, -output 指定输出文件
- v, -verbose 详细模式
- n, -dry-run 不做任何改变
- i, -interactive 覆盖前先询问
- openpgp 行为严格遵循 OpenPGP 定义

-pgp2

生成与 PGP 2.x 兼容的报文

(请参考在线说明以获得所有命令和选项的完整清单)

范例：

-se -r Bob [文件名] 为 Bob 这个收件人签字及加密

-clearsign [文件名] 做出明文签字

-detach-sign [文件名] 做出分离式签字

-list-keys [某甲] 显示密钥

-fingerprint [某甲] 显示指纹