docboon

GnuPG

GnuPG(gpg)使用指南

加密、数字签名和非对称密钥

https://docboon.github.io/

GnuPG(gpg)使用指南: 加密、数字签名和非对称密钥

https://docboon.github.io/
Mr. Older Flower

出版日期 September 28th,2021

修订历史		
修订 2.0	2021年10月18日.	<olderflower @outlook.com>
更改插图		
修订 1.2	2021年8月16日.	<olderflower @outlook.com>
首个版本		

内容简介绍

本文源自https://www.gnupg.org/howtos/，中文版由Rui Liu<rui_liu_1@yahoo.com>翻译。为了更清晰地了解gnupg的使用，docboon在原文的基础上进行了配图，并添加“情景模拟”章节。

本文对非对称加密做了简单的介绍；并精炼地介绍了gnupg的使用。

目录

插图清单

排版约定

在正文中会有一些诸如程序代码、系统命令或是屏幕输出一类的信息，为了能清晰地展示各种元素，本手册遵守下列排版约定。

1. **字体约定**，在正文中嵌入的一些系统命令、文件名、函数或是参数等使用有别于正文的字体：

   a. 等宽字体：程序片段、正文中出现的配置选项、变量、函数名等，我们采用等宽字体，样式如下：`serverip`。

   b. 等宽微粗字体：表示由用户输入的系统命令。例如，在linux下，我们查网络接口信息的命令：`ifconfig eth0`。

   c. 等宽斜体：表示由用户输入的值或是一些需要设定的参数的值，例如，用户在浏览器里输入的一个url，用如下的样式显示：*`http://docboon.io`*。

   d. 等宽微粗斜体字体：表示文件名、数据库名及新的术语等。例如，文件的名称用如下的样式来显示：***`myfirst.xml`***。

2. **提示、告诫类信息**（admonitions），有一些内容是正文的补充或是对用户的一种提醒。为了能清晰地展示这部分内容，采用独立的“信息块”来呈现。一共有5种，分别为：

   - 

     提示

     可以使用`apt-get upgrade`来升级您的ubuntu或debian Linux系统。

   - 

     注意

     在运行`apt-get upgrade`之前，需要先运行用`apt-get update`命令。

   - 

     重要

     请及时升级系统补丁程序，这是保证系统处于安全状态的好方法。

   - 

     小心

     如果系统升级还没有完成，请不要重新启动您的电脑。

   - 

     警告

     在系统升级过程中千万不要关闭电源。

3. **屏幕内容围栏及程序代码围栏：**

   - 屏幕内容围栏，有时候我们需要整块显示在屏幕上输出的内容，或者是键入的命令。例如，在linux下，我们使用`ifconfig eth0`命令来查网络接口信息：

```
$ ifconfig eth0
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 98:fa:9b:db:b9:9f  txqueuelen 1000  (以太网)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
```

```
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 16  memory 0xea300000-ea320000
```

- 程序代码围栏，对于一些程序代码，使用带有序号的样式来展示，例如，一段php代码：

```
 1 <?php
 2   class SimpleBook {
 3     private $title;
 4
 5     function __construct($title_in) {
 6       $this->title  = $title_in;
 7     }
 8
 9     function getTitle() {return $this->title;}
10   }
11 ?>
```

第 1 章 概念

1. 公钥加密

加密的传统方法是只用一把密钥加密。信息发送者用这把钥匙对信息加密。接收信息者需要有完全相同的钥匙才能将加密了的信息解密。这把钥匙必须以一种其他人没有机会得到它的方式给予接收信息者。如果其他人得到了这把钥匙，这种加密方式就没用了。

使用一种称为“公开钥匙”的方法可以解决这个问题。公开钥匙的概念涉及两把钥匙。

1. **公钥**：一把钥匙称为公钥（公开钥匙），可以以所有方式传递，任何人都可以得到。公钥通常有如下两个作用：
   - 信息发送者使用信息接收者的公钥对信息加密，然后将信息发送给接收者；
   - 信息接收者利用信息发送者的公钥验证信息的发送人身份。
2. **私钥**：另一把钥匙称为私钥（隐密钥匙），这把钥匙是秘密的，不能传递出去。只有它的拥有者才能接触和使用它。私钥通常有如下两个作用：
   - 信息接收者使用自己的私钥解密信息；
   - 信息发送者用自己的私钥对要发送的信息签名。

如果正确实施了这种方法，从公钥不能得出密钥。信息发送者以信息接收者的公钥将信息加密，接收者则以自己的密钥解密。

这个概念的关键之处在于密钥必须保持秘密，不能随便给出或让任何除了密钥拥有者之外的人得到。请千万不要将你的密钥通过Internet寄出！另外，通过telnet使用GnuPG是非常不明智的。

2. 数字签名

为证明一则信息确实是宣称信息发送者的人所发，发明了数字签名的概念。正如其名称显示，信息发送者数字化地在信息上签名。别人可以通过这个签名检验这个信息的真实性。使用这种方法，可以减少中“特洛伊木马计”的风险（即一则信息宣称是对某个问题的补丁，实际却包含病毒或乱动你计算机上的数据），同时信息或数据可以被确认是来自正当合法的来源，而被认为属实。

一个数字签名是通过密钥和信息本身而得来。信息可以通过信息发送者的公钥来验证。这样，不仅可以验证信息是正确的信息发送者所发，而且内容也得到验证。这样，信息接收者可以确认：信息来自该信息发送者，而且在传递过程中其内容没有改变。

3. 信任网

公开钥匙算法的一个弱点在于如何传播公开钥匙。有可能有用户传递一把有虚假身份的公开钥匙。如果别人不知就里，用这把公钥加密信息，持有该虚假钥匙的侵入者就可以解密而读到信息。如果侵入者再将解密的信息以真正的公开钥匙加密，然后传送出去，这种进攻无法被发现。

对此问题，PGP的解决方法（因此也自动是GnuPG的解决方法）是对公开钥匙签名。每把公开钥匙都有一个相应的用户身份。一个人的公开钥匙可以由别人来签名。这些签名承认这把钥匙确实属于它所宣称的用户。至于有多信任这些签名，完全取决于GnuPG用户。当你信任给这把钥匙签名的人时，你认为这把钥匙是可信的，并确信这把钥匙确实属于拥有相应用户身份的人。只有当你信任签名者的公开钥匙时，你才能信任这个签名。要想绝对确信一把钥匙是正确和真实的，你就得在给予绝对信任之前，通过可靠渠道比较钥匙的“指纹”。

4. 指纹

如何确定一把公钥是属于某个人的？每把公钥都有唯一的指纹，例如，当获得Bob的公钥时，要确定这把公钥确实是属于Bob的，那就要比对这把公钥的指纹，这个过程通常是通过电话或见面的方式确定的。

5. 安全边界

如果你有数据想要保密，你所需做的远不止选择加密算法这一件事。你应该统筹考虑你的系统安全。一般我们认为PGP是安全的。在作者写本文时，尚未听说任何PGP被破译的事例。但这并不表示所有用PGP加密的信息都是安全的（举例说，如果NSA--美国国家安全局破解了PGP，它绝不会通知我。别的为真正邪恶目的破译密码的人也不会）。反过来说，即使PGP是完全“无法破译”的，也可以用别的方法来损害安全。

另一种可能的技术（虽然更难做到）是使用一种“特洛伊木马”程序，它可以传出用户所敲的键。也可以（但非常困难）传出屏幕显示的内容。使用这些技术，就根本不需要破译加密的信息了。针对以上这些危险，需要制定一个好的，深思熟虑的安全计划并付诸实施。

提到上述这些，目的并非想让人们怀疑一切，而是想指出需要采取很多措施才能达到更安全。最重要的是意识到加密只是安全的一个步骤，而不是全部的解决方案。

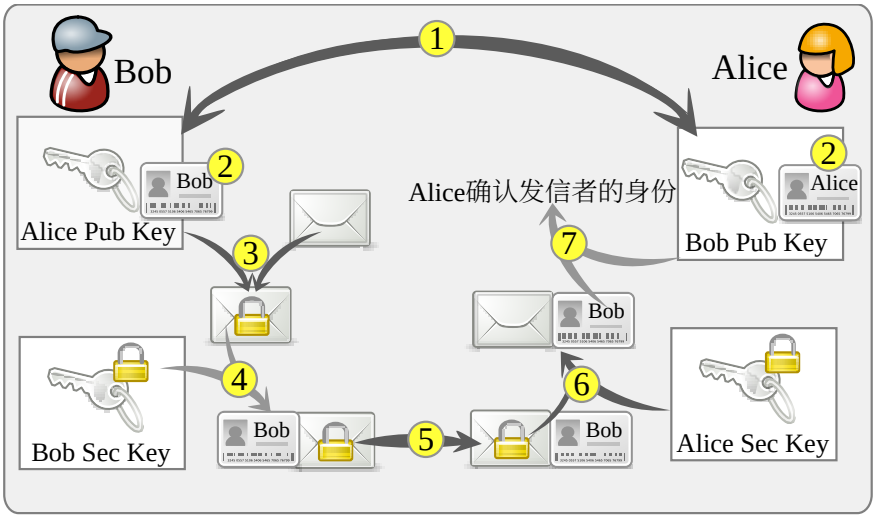
6. PGP用户互联组

用户可以通过对相互公钥进行签名建立一个PGP用户互联组。比如Bob认识Alice，Bob将自己的公钥传给Alice之后，Alice在Bob的公钥上签名并传回给Bob，Bob保存这份签名的公钥。当Bob和Eve通信时，就将有Alice签名的自己的公钥复制给Eve，Eve之前已经有了Alice的公钥并信任Alice签名的其他用户的公钥，所以当他用Alice的公钥验证该签名有效时，就接收Bob，这样Alice就将Bob介绍给了Eve。

7. GnuPG处理过程

在明白了上述的一些基本概念后，我们可以用下图表示“公开钥匙”的处理过程：

图 1.1. 处理过程



1：Bob和Alice交换公钥；

2：Bob和Alice通过指纹确认对方的公钥是信任的，并在上面签名；

3：Bob利用Alice的公钥和要发送的信息生成加密信息；

4：Bob利用自己的私钥对加密信息签名；

5：Bob发送信息给Alice；

6：Alice利用自己的私钥解密信息；

7：Alice利用Bob的公钥验证签名信息以确定这个信息就是Bob发送的。

第 2 章 安装

1. GnuPG的源程序

正式的下载站点是：GnuPG主页 [http://www.gnupg.org/download.html]。

由于法律限制的缘故，从美国的服务器下载GnuPG是不被允许的。美国对加解密软件的出口有限制。这是为什么PGP总是有国际和美国两个版本的原因。国际版的源程序是以印刷成书的方式出口的。书到了欧洲（挪威奥斯陆）再被扫描进计算机。更多这方面的信息可以在国际PGP主页找到。PGP的国际版可以自由进入美国，只要它不又被出口出去。

2. 设置

你可以用如下方式得到GnuPG：Debian软件包，RPM（Redhat软件包管理程序）软件包，或源程序。这些软件包将在Linux系统上安装执行文件及所需的工具。若你想在别的系统上使用GnuPG，则需要自己编译源程序。

由于该程序的主要部分是在Linux（x86）操作系统上进行的，将它移植到别的操作系统上应该不会太难。已知的支持GnuPG的操作系统的名单可以在GnuPG主页找到。下列设置方法适用于大多数系统，它用于从一个源程序打包来安装GnuPG:

先用以下命令打开该打包：

```
tar xvzf gnupg-x.x.x.tar.gz
```

然后，cd到包含源程序的目录。然后敲入

```
./configure
```

做这件事时，应该没有任何特殊的事情发生。如果敲入

```
./configure --help
```

你可以看到编译时的所有可选项。如果出现与国际化（GET　　text）有关的问题，你可以用--with-included-gettext这个选项包含一个与源程序同在一个打包中的版本，或用--disable-NLS选项来将其关掉。

3. 编译

这之后可以敲入下列来编译：

```
make
```

这一步应该不会出任何问题。如果出了问题，请按照顺序采取如下步骤：首先，试试能否自己解决问题（当然要参考可以弄到的各种档案资料）。然后，确定这个问题是否是一个已知的软件错误--bug（请检查http://www.gnupg.org处的BUGS文件）。 如果问题与路径有关，你应试试先make clean，然后再次运行configure和make。

4. 安装

现在敲入：

```
make install
```

来真正将可执行程序和手册页拷贝到安装目录中。如果你在设置./configure时没有更改安装目录，那么隐含的安装目录将是*/usr/local/share/gnupg/*。你可以在文件*options.skel*中找到这个目录。如果你把这个文件拷贝成*~/.gnupg/options*文件并对其中的选项做出适当的改变，那么这个文件将被用作标准。当建立*~/.gnupg/*目录时，上述文件的拷贝将自动进行。在该文件中，所有选项均有良好的说明资料，在此解释它们意义不大。

你可以以设置有效用户身份为超级用户（suid root）的方式来运行GnuPG，这样程序运行时就有超级用户的所有权限。这样做，你就去除了一种危险，即程序的一部分被存储于内存之外而可能被别人读到。由我来帮助分析这样做的风险是不现实的。但是，以suid root的方式来执行这个程序，你应该警惕特洛伊木马的危险，因为一个以超级用户权限运行的特洛伊木马有能力破坏整个系统。如果因为这个原因（或任何其他原因）你选择不以超级用户权限来运行GnuPG，你可以在*~/gnupg/options*文件中设置no-secmem-warning来关掉相应的警告信息。

第 3 章 使用钥匙

1. 产生钥匙

使用以下命令

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

以产生一个新的钥匙对（一个钥匙对包括一把隐密钥匙和相应的一把公开钥匙）。运行时，第一个问题是选择使用哪个算法。你可以在 PGP DH vs. RSA FAQ(http://www.hertreg.ac.uk/ss/pgpfaq.html) 或 Applied Cryptography (http://www.gnupg.org/(en)/howtos/ch/GPGMiniHowto-6.html#BSchneier)中读到更多算法方面的信息。对这个问题可以很容易地选择DSA/ElGamal。该算法没有专利。

下一个问题是钥匙长度。对这个问题的答案因人而异。你需要在安全和计算时间二者选一。如果钥匙长些，讯息被截获后被破解的风险就小些。但是使用时计算时间也要长些。如果计算时间是考虑的重要因素，你仍然应该考虑到你将使用这把钥匙一段时间。我们都知道由于新处理器越来越快，使得数值计算速度增加非常快。因此，选择钥匙长度时，要考虑到这个因素。GnuPG要求的最短钥匙长度是768字位。不过，有人认为你应该使用2048位的钥匙（这实际也是GnuPG目前允许的最大长度）。就DSA而言，1024位是一个标准的长度。当安全是首要考虑因素而速度相对次要时，你应该选用最大可能的长度。

程序现在要你输入你的名字，介绍和电子邮件地址。密码将根据以上输入的内容产生出来。你可以在将来改变这些设置。见钥匙管理。你应该用一个较长的真实的电子邮件地址，因为以上完整的用户身份识别（包括名字，介绍和电子邮件地址）都被签名。如果你修改了地址，签名就将与修改后的内容不符。最后一点，你可以输入一些介绍性内容。在此你输入什么都行，可以相当有用。

最后，你得敲入密码，或称通行字（实际上，由于允许使用空格，叫作通行句更合适）。该密码使你可以使用与你的密钥有关的各项功能。一个好的通行句具有下列要素：

- 长度要长，
- 包含特殊字符（非英文字母或阿拉伯数字），
- 要是某种特殊的东西（不能是姓名），
- 要很难被猜到（因此请万勿使用姓名，生日，电话号码，信用卡号码，支票帐号号码，小孩的姓名或人数等）

在通行句中，用大小写混杂的方式你可以进一步增进安全。当你想出密码时，一定要保证你不会忘掉它。如果你忘掉了，发给你的讯息就再也没法解密因而再也没法读了，你的密钥也再也用不成了。产生包含这个信息的某种形式的证书可能是一种明智之举。（当然要小心不要让别人得到你的通行句）。见取消钥匙.

在上述所有都输入之后，系统开始产生钥匙。这会花一些时间。在这段时间，它需要收集许多随机数据。如果你在这段时间在另一个窗口工作，你就可以帮助系统收集随机数据。正如你现在已经懂得的，产生出的钥匙总是不同的。如果你在现在和五分钟后输入完全相同的数据来产生钥匙，你会得到两把不同的钥匙。现在你一定明白为什么你不应该忘记你的通行句了。

2. 输出钥匙

下列命令可以输出一个用户的钥匙：(输出的是公钥还是密钥？应该是公钥)

```
gpg --export [UID]
```

如果没有给出一个用户身份识别（UID），所有现有的钥匙都会被输出。隐含地，结果将输出到标准输出（ stdout）去。但是用-o 选项可以把它放到一个文件里去。建议你同时使用 -a 选项来把钥匙写进一个7位的文件，而不是一个二进制文件。

输出钥匙后，你就可以增大你的视野。别人可以与你安全地通讯了。你可以把钥匙放在你的主页上，用finger服务，用钥匙服务器，或任何别的你能想到的方式来让别人知道你的公钥。

3. 引进钥匙

当你收到一把别人的公钥（或好几把公钥）时，为了能使用它们，你得把它们加进你的钥匙数据库。加进数据库的命令如下：

```
gpg --import [Filename]
```

如果文件名（filename）省略了，数据将从标准输入（stdin）读入。

4. 取消钥匙

因为好些原因，你可以想要取消一把已经存在的钥匙，例如：密钥被盗了或被不该得到它的人得到，用户身份识别改变了，钥匙不够长了，等等。对上述各种情况，取消钥匙的命令是：

```
gpg --gen-revoke
```

该命令将产生一份取消钥匙证书。 要这么做，一定要先有密钥！ 否则任何人都能取消你的钥匙。这种方法有一个缺点：如果我不知道通行句就用不了密钥。但用不了密钥，我就不能取消我的钥匙。为解决这个问题，在你产生钥匙对的时候就产生一份取消钥匙证书是一种明智的做法。如果你这样做的话，一定要把证书保存好！你可以把它放在磁盘上，纸张上，等等。一定要保证证书不落入坏人之手！！！否则别人就可以发出该证书取消你的钥匙，使你的钥匙作废。

5. 钥匙管理

随系统而来，有一个文件，起到某种数据库的作用。所有有关钥匙和钥匙附带信息的数据都存在这个文件里（只有一样例外：主人的信任值。更多的信息见钥匙签名）。用

```
gpg --list-keys
```

可以显示所有现有的钥匙。要想同时显示签名，用

```
gpg --list-sigs
```

（更多的信息见 钥匙签名）。要想见到钥匙的指纹，敲入：

```
gpg --fingerprint
```

用户需要见到”指纹”来确认某人的确是其自称是的人（就象在电话中一样）。这个命令将会产生一列相对较小的数字。

要列出私密钥匙，你可以敲

```
gpg --list-secret-keys
```

注意：列出私密钥匙的指纹和签名根本就没用。

要删除一把公钥，你可以敲

```
gpg --delete-key UID
```

要删除一把密钥，你可以敲

```
gpg --delete-secret-key
```

还有一个与钥匙有关的重要命令：

```
gpg --edit-key UID
```

用此命令你可以修改钥匙的失效日期，加进一个指纹，对钥匙签名等等。尽管显得太清楚而不用提，这里还是要说，要做以上事情你得用你的通行句。敲入通行句后，你会见到命令行。

6. 钥匙签名

正如前言所提到的，这个系统有一个最大的薄弱点，那就是公钥的真实性问题。如果用的是错误的公钥，你加密的价值就全没了。要克服这种风险，一种可能是对钥匙签名：你把你的签名放在一把钥匙上，这样你就绝对确信这把钥匙是真实的。这样，钥匙上的签名就表示承认钥匙上的用户身份确实是这把钥匙的主人。有了这个保证，用户就可以开始放心用这把钥匙加密了。

要对一把钥匙签名，用 `gpg --edit-key UID` ，然后用sign命令。

你只有在绝对确信一把钥匙的真实性的时候，才应该对它签名认可！！！ 例如你本人拿到了这把钥匙（比如说在一个钥匙签名派对上），或者你通过别的渠道得到了钥匙，然后检查了它的指纹（例如通过打电话询问）。你永远不应该光凭假设就对一把钥匙签名。

GnuPG根据现有的签名和”主人的信任度”来决定钥匙的真实性。主人信任度是钥匙的主人用来决定对别的某把钥匙的信任程度的一个值。这个值可以是

- 1 = 我不知道
- 2 = 我不信任（这把钥匙）

- 3 = 我勉强信任
- 4 = 我完全信任

如果用户不信任一个签名，可以就这么说，因而将弃这个签名不用。这些信任信息不是存在储存钥匙的文件里，而是存在另一个文件里。

第 4 章 加密和解密

在安装和按照需要设置好所有事以后，我们就可以开始加密和解密了。

加密解密时，可以使用超过一把隐密钥匙。如果这种情况出现，你需要选择其中一把作为活跃的钥匙。要这样做，可以用命令行选项 -u UID 或用 --local-user UID 。这样做将以我们想要使用的钥匙取代隐含的钥匙。

如果你想改变讯息接收者，可以用命令行选项 -r 或 --recipient 。

1. 加密

加密的命令是：

```
gpg -e Recipient [Data]
```

或

```
gpg --encrypt Recipient [Data]
```

为避免别人宣称是你的风险，对所有你加密的东西签名是有用的。见 签名。

2. 解密

解密的命令是：

```
gpg [-d] [Data]
```

或

```
gpg [--decrypt] [Data]
```

在此，预设输出为 stdout ，但可以使用 -o 选项将输出转向到一个文件。

第 5 章 签名和检验签名

以下命令可用于以你的钥匙对数据签名：

```
gpg -s (--sign) [Data]
```

这样做的时候，同时数据也被压缩。也就是说，最终结果是无法直接读懂的。若你想要一个能直接读懂的结果，你可以用：

```
gpg --clearsign [Data]
```

这样就能保证结果是清晰可读的。同时它也照样对数据签名。

用

```
gpg -b (--detach-sign) [Data]
```

你可以将签名写进另一个文件。我们高度推荐这种用法，尤其是对二进制文件（如文档）签名的时候。另外， --armor 选项在这儿也非常有用。

经常你会发现有些数据既加了密又签了名。要这么做，完整的命令行大致如下：

```
gpg [-u Sender] [-r Recipient] [--armor] --sign --encrypt [Data]
```

选项-u (--local-user) 和 -r (--recipient) 的作用如前所述。

如果数据既加了密又签了名，签名是在解密过程中检验的。你可以用以下命令检验签名：

```
gpg [--verify] [Data]
```

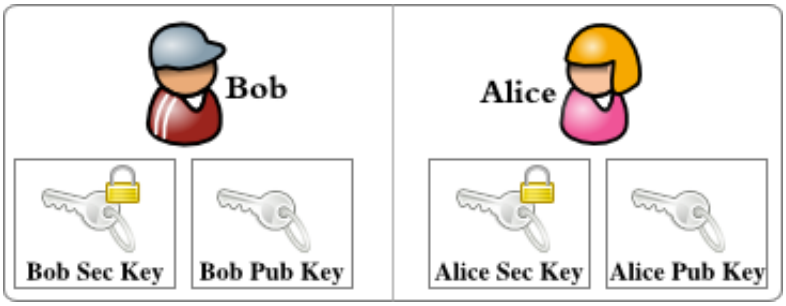
当然，只有当你有讯息发出者的公钥时，这才起作用。

第 6 章 情景模拟

1. 产生密钥对

Bob和Alice在自己的机器上使用gpg --gen-key命令产生各自的密钥对，一把公钥和一把私钥，其中，公钥保存于*~/.gnupg/pubring.gpg*公钥环中，私钥保存于*~/.gnupg/secring.gpg*私钥环中。

图 6.1. 产生钥匙对



可以通过gpg --list-keys来查看公钥，通过gpg --list-secret-keys来查看私钥。

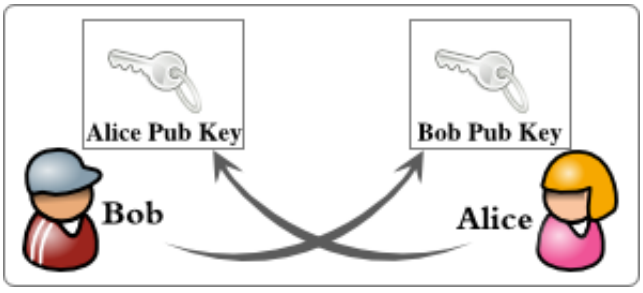
最后Bob通过 gpg -a -o bob.asc --export bob@lognovel.com 导出自己的公钥*bob.asc* 。

Alice通过同样的方式产生自己的密钥对，并导出自己的公钥*alice.asc* 。

Alice和Bob已经可以交换各自的公钥并进行加密的通信了。

2. 交换公钥

图 6.2. 交换公钥



Alice和Bob首先要获取对方的公钥，获得对方的公钥通常有如下的途径：

1. **通过互联网**

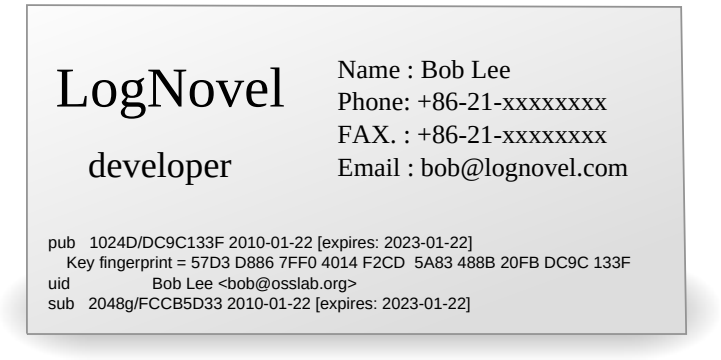
可以是电子邮件也可以是http或ftp下载，获得对方的公钥后，首先要将其导入到自己的公钥环中，例如，Bob获得了Alice的公钥后使用 `gpg --import alice.asc` 命令将其导入。

双方通过 `gpg --fingerprint` 命令来查看*alice.asc*的指纹，然后通过电话或双方认为安全的方式对指纹进行比对，用以确定*alice.asc*确实属于Alice。

2. **在黑客/开发人员聚会上**

在黑客/开发人员聚会上人们除了会交换自己的公钥外，更多的是交换各自的密钥指纹，通常情况下，很多黑客或开发人员会将自己的密钥指纹印制在名片上，下图是一张印刷了指纹的名片。

图 6.3. Name Card



在黑客/开发人员在出示自己的指纹或交换印刷有指纹的名片时，还要出示自己的身份证、护照、学生证、士官证或其他有效的身份证件让对方进行查验。

提示

如果你要参加重要的黑客/开发人员会议（例如您参加lognovel开发人员的聚会），准备一张印刷有自己密钥指纹的名片是非常重要的，在这样的会议上通常会有较为正式的密钥指纹交换仪式，大家相互交换指纹并出示自己的有效身份证件，并相互查验。

3. 对公钥签名

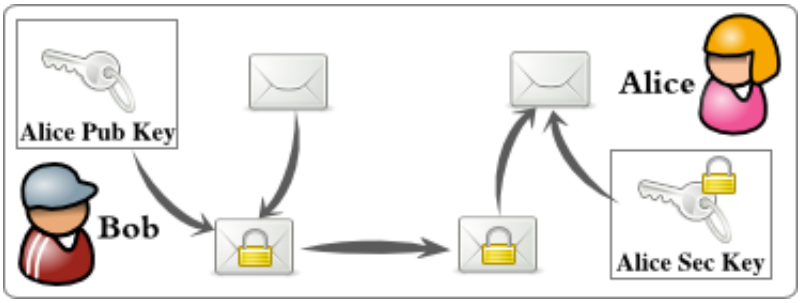
Bob在确定*alice.asc*确实属于Alice后，便可以在Alice的公钥上签名，使用

```
gpg --edit-key alice@lognovel.com
```

对Alice的公钥进行编辑，进入命令行以后再使用sign子命令进行签署。

4. 发送信息

图 6.4. 发送信息



现在Bob已经有了Alice的公钥，并对其进行了确认，接下来，Bob就可以用Alice的公钥对信息进行加密，并将加密的信息发送给Alice了。

1. **Bob对信息进行加密**

   Bob有一个程序文件*code.php*需要发送给Alice，Bob使用如下的命令对其进行加密：

   ```
   gpg -e code.php
   ```

   需要输入收件人的UID `Alice Lee`或`alice@lognovel.com`

   之后会生成一个新的文件*code.php.gpg*，Bob将这个加密后的文件发送给Alice。

2. **Alice对信息进行解密**

   Alice在收到*code.php.gpg*后使用

   ```
   gpg -o code.php -d code.php.gpg
   ```

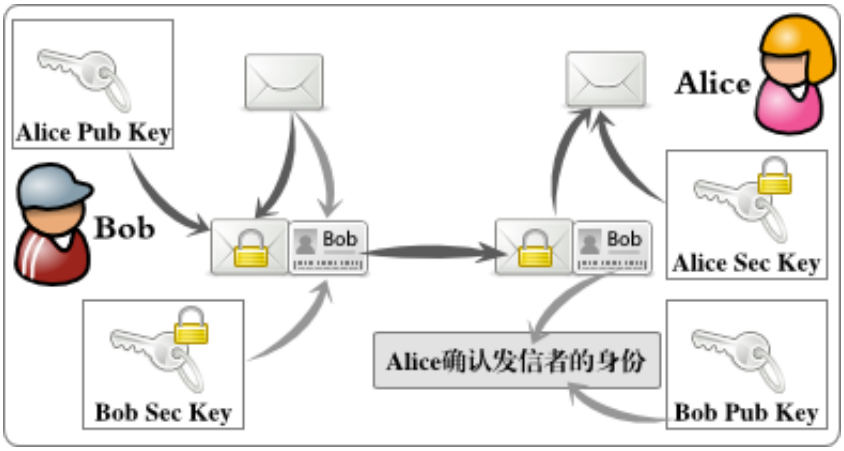
   命令进行解密，在解密过程中Alice需要输入在生成密钥时输入的“通行句”，正常解密后会生成*code.php*文件，这个文件的md5码和Bob机器上的*code.php*的md5码是一样的。

5. 对发送的信息签名

前面Bob使用Alice的公钥对*code.php*进行了加密，并发送给了Alice，实际上，从Alice的角度来讲还是有风险的，因为Alice在收到*code.php.gpg*后，虽然可以使用自己的私钥解密，但她并不能确定这个文件一定是Bob发送给他的。Alice如何确定这个文件是Bob发送的呢？这就需要Bob多做一件事：对要发送的*code.php*进行数字签名，然后Alice用Bob的公钥对其进行确认。

1. **将加密文件和签名信息放在一个文件中**

图 6.5. 将加密文件和签名放在一个文件中




```
gpg -u bob@lognovel.com -r alice@lognovel.com -s -e code.php
```

这个命令将会生成*code.php.gpg*文件，这个文件中已经包含了Bob的签名信息。当Alice收到这个文件后使用

```
gpg -d code.php.gpg
```

会清晰的看到原文件和Bob的签名信息，前提是Alice一定要有Bob的公钥。

2. **将源文件和签名信息分离**

```
gpg -b code.php
```

这个命令将会生成*code.php.sig*文件，这个文件中已经包含了Bob的签名信息。当Alice收到*code.php*和*code.php.sig*文件后使用

```
gpg --verify code.php.sig
```

来验证发送者的身份。

```
gpg -a -b code.php
```

这个命令将会生成*code.php.asc*文件，这个文件中已经包含了Bob的签名信息。当Alice收到*code.php*和*code.php.sig*文件后使用

```
gpg --verify code.php.asc
```

来验证发送者的身份。

第 7 章 信息来源

1. GnuPG

- GnuPG主页：http://www.gnupg.org/
- GnuPG电子邮件名单，包括 GnuPG主页(http://www.gnupg.org/docs.html) 上的旧文档和描述。
- GnuPG项目中所包含的信息，虽然还没有做很多。另外别忘了用

```
gpg --help
```

gpg的帮助信息。

2. PGP

PGP是一个较老的，仍然被广为使用的加解密软件。自其出现后的这些年里，人们为其写了很多辅助资料。这些都是十分有用的信息。其中许多涵盖广泛，对GnuPG也适用。请在以下URL查找这些资料：

- 国际PGP主页：https://www.openpgp.org/
- PGP DH和RSA算法 (http://www.hertreg.ac.uk/ss/pgpfaq.html)比较常见问题 包含这两种算法区别的信息。GnuPG正是使用这两种算法。