 互动：数据分析与云计算应用案例（征集|参与）

QuietHeart

quietheart.blog.chinaunix.net

Find the things you want.

[首页](#) | [博文目录](#) | [关于我](#)

vaqeteart

博客访问：2479871

博文数量：1029

博客积分：19333

博客等级：上将

技术积分：11053

用户组：普通用户

注册时间：2007-03-27 14:33

加关注

短消息

Linux命令学习手册-gpg命令

2011-06-28 19:32:14

分类：LINUX

gpg

[功能]

GPG是加密和数字签名的免费工具，大多用于加密信息的传递。除了仅用密码加密外，GPG最大的不同是提供了“公钥/私钥”对。利用你的“公钥”别人加密信息不再需要告诉你密码，随时随地都能发送加密信息。而这种加密是单向的，只有你的“私钥”能解开加密。数字签名又是另一大使用方向。通过签名认证，别人能确保发布的消息来自你，而且没有经过修改。

[原理]

对称密钥加密常用的算法DES、Triple DES或IDEA加密，MD5。对称密钥加密通信双方持有同样的密钥对密文进行解密。

文章分类

全部博文 (1029)

为知笔记 (0)

+ 记事相关 (32)

+ 工作求职 (28)

+ 其他资料 (145)

+ 生活百科 (75)

+ 学习相关 (500)

+ 娱乐休闲 (12)

个人随笔 (123)

+ Tmp&Test (21)

未分配的博文 (93)

文章存档

+ 2015年 (11)

+ 2014年 (19)

+ 2013年 (51)

+ 2012年 (329)

+ 2011年 (292)

+ 2010年 (106)

+ 2009年 (57)

+ 2008年 (164)

我的朋友

公共密钥加密在报文和网络方面的应用已成为流行。RSA实际上用于公共密钥加密，它是当前所能得到的最强的公共密钥算法。公共密钥包含两个同属于一方的密钥：一个是公共密钥，它被所有人所共享；另一个是私有密钥，归个人秘密中存。与对称密钥加密不同，公共密钥加密对加密与解密使用两把密钥。一把是秘密的，这是私有密钥，用来对密文解密。密文本身由公共密钥产生，公共密钥分发给要发送加密信息给你的人。其他人如何得到你的公共密钥？很容易，发布它即可。你——私有密钥的所有者，是惟一能对信息进行解密的人。

数字签名常用公共密钥加密来产生签名，如RSA和DSA。用公共密钥产生数字签名与信息加密的工作相反。信息通过hash函数发送。hash处理后的信息用私有密钥加密。一旦数据用私有密钥加密，任何持有公共密钥的人都能检验，它是用私有密钥产生的，这样，数据就被验证了。因此，任何持有公共密钥的人都可以进行检验。这里的hash函数用于产生数字签名。Hash函数(散列函数)是一个提取信息和产生一个固定长度的信息个性特征的数学函数。无论要进行hash方法的信息有多大，输出的长度都一样。简言之，就是将hash处理后的信息用私钥加密，其它人用公钥解密恢复成功(恢复成的应该是hash处理的结果?)表示加密的人就是你也就是数字签名的验证就成功了。

[举例]

****对称加密和解密**

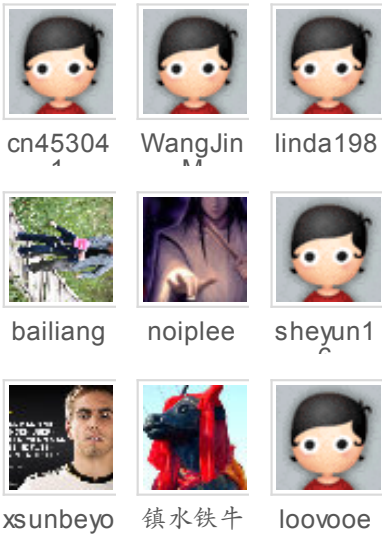
这里，先简单介绍使用gpg进行对称加密和解密，后面重点介绍公钥加密和数字验证。

对称加密myfile：*#gpg -c myfile**

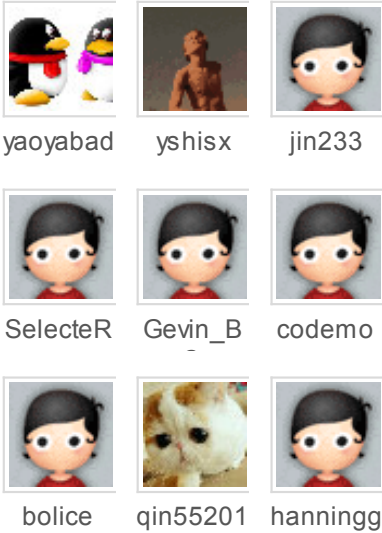
这样，会提示你输入两次密码，然后生成myfile的加密文件myfile.gpg,注意不能对目录加密,可以用-o选项指定输出文件名称。

***解密对称加密的文件：**

```
#gpg -o mydecrypt -d myfile.gpg
```



最近访客



微信关注



IT168企业级官微
微信号：IT168qiye

gpg --gen-key

这样，会提示你输入解密密码，输入之后，将会解密相应的文件，通过-o指定生成的解密文件，-d指定被解密的文件。

**

**公钥加密相关

*生成钥匙对：

[root@lv-k ~]# gpg --gen-key

输入之后，输出和交互提示如下所示：

#####以下为输出以及交互#####

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.

gpg: 已创建目录‘/root/.gnupg’

gpg: 新的配置文件‘/root/.gnupg/gpg.conf’已建立

gpg: 警告：在‘/root/.gnupg/gpg.conf’里的选项于此次运行期间未被使用

gpg: 钥匙环‘/root/.gnupg/secring.gpg’已建立

gpg: 钥匙环‘/root/.gnupg/pubring.gpg’已建立

请择您要使用的密钥种类：

(1) DSA 和 ElGamal (默认)

(2) DSA (仅用于签名)



系统架构师大会

微信号：SACC2013

订阅



推荐博文

- Docker+Redis3集群环境搭建...
- LVM 类型的 Storage Pool - ...
- 自动化运维之cobbler批量部署...
- Redhat下puppet集中配置管理...
- 使用flume+kafka+storm构建实...
- Oracle中常见的33个等待事件...
- 一个oracle bug的简单验证...
- RAC共享磁盘物理路径故障导致...
- Hadoop和Spark有啥区别
- 【ERROR】非DBA用户要使用aut...

热词专题

- 移动硬盘数据恢复
- dual 在plsql函数中的影响...
- MongoDB 索引要点
- 欢迎魄力中国在ChinaUnix博客...

(5) RSA (仅用于签名)

您的择？[Enter] <====输入

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)[Enter] <====输入

您所要求的密钥尺寸是 2048 位

请设定这把密钥的效期限。

0 = 密钥永不过期

= 密钥在 n 天后过期

w = 密钥在 n 周后过期

m = 密钥在 n 月后过期

y = 密钥在 n 年后过期

密钥的效期限是？(0)[Enter]

密钥永远不会过期

以上正确吗？(y/n)y <====输入

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) ”

真实姓名：quietheart <====输入

电子邮件地址：quiet_heart000@126.com <====输入

注释：test <====输入

您定了这个用户标识：

“quietheart (test) ”

更改姓名(N)、注释(C)、电子邮件地址(E)或确定(O)/退出(Q)? 0 <====输入

您需要一个密码来保护您的私钥。

请输入密码：<===输入

请再输入一次密码：<====输入

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。

[illegible]

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。

[illegible]

gpg: /root/.gnupg/trustdb.gpg: 建立了信任度数据库

gpg: 密钥 DDBA2DEA 被标记为绝对信任

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： 1 已签名： 0 信任度：0-，0q，0n，0m，0f，1u

pub 1024D/DDBA2DEA 2011-06-14

密钥指纹 = 790A 0F2D 6826 61F3 A749 0724 DBB2 C0A5 DDBA 2DEA

uid quietheart (test)

sub 2048g/2BBE2C91 2011-06-14

#####以上为输出以及交互#####

这里，首先要一个钥匙对才能公钥加密，这里根据系统情况不同，可能内容为英文。需要用户交互输入的地方，都通过"<===输入"这个标记指出来了。若对输入信息有所改动，可把 ~/.gnupg 目录下除 options 以外的文件删除，再运行 gpg --gen-key 命令，或者使用 gpg 的 edit 选项。钥匙对放在 ~/.gnupg 目录下。

***查看已有的钥匙：**

[root@lv-k .gnupg]# gpg --list-keys

输入之后，输出如下：

#####以下为输出以及交互#####

/root/.gnupg/pubring.gpg

pub 1024D/DDBA2DEA 2011-06-14

uid quietheart (test)

sub 2048g/2BBE2C91 2011-06-14

#####以上为输出以及交互#####

这里，

pub(公匙)--- public key , ID : DDBA2DEA

sub(私匙)--- secret key or private key , ID : 2BBE2C91

假如没使用root进行操作，可能会输出如下信息："gpg: Warning: using insecure memory!" 警告没有锁定内存页，一般是连接网上操作有关安全方面的问题，没有事情的，可以作这样的改动 "# chmod 4755 /usr/bin/gpg"。

*使用gpg密钥进行加密和解密文件：

下面我们通过一个具体的例子，演示公钥加密，私钥解密的过程。这里加/解密方式采用RSA算法，公匙与私匙是互补，理论上是不可破解，也没有人尝试成功过。假设我们已经使用前面的方法生成了密钥对。

1，首先查看待加密的文件如下：

```
[root@lv-k gpg_test]# ls
```

```
mygpgtest
```

```
[root@lv-k gpg_test]# cat mygpgtest
```

```
#####以下为输出#####
```

```
hello!
```

```
welcome come to here
```

```
today is 2011-06-14
```

```
#####以上为输出#####
```

2，公钥加密过程

1)使用如下命令加密文件：

```
[root@lv-k gpg_test]# gpg -ea -r quietheart mygpgtest
```


这里，使用的密钥就是前面创建的quietheart，我们可以使用"gp -list-keys"来查看我们可以选择用来加密的公钥。这里选项"-e"实际就是"--encrypt"表示加密数据；"-a"表示创建ASCII的输出(可以不用这个选项,这样生成的文件就不是ASCII的内容了，并且文件后缀是*.gp)；"-r"指定加密的用户id名称。

2)查看加密之后的文件：

```
[root@lv-k gp_test]# ls
```

```
mygpptest mygpptest.asc
```

```
[root@lv-k gp_test]# cat mygpptest.asc
```

```
#####以下为输出#####
```

```
-----BEGIN PGP MESSAGE-----
```

```
Version: GnuPG v1.4.5 (GNU/Linux)
```

```
hQIOA6PEEmMrviyREAf9HCZ5xIIspneZ6i7Hquxb7xUjn1q0W5ccVek6x0DxSbH
q55ugy6CmCc/excLC/zblf9qHsNDcZvMV3jjD95gu78NR1lsyDtpG9r5bX/MuPii
KxYc3oOvGNmDUO9F/g3ul9VCu/rsIkQvwXZHaEGXR3G6XH/tmhKZcjNNIxb1qQiq
xE7O0NCXEhp8FpOPozY1MzZ7wv4rXLujTrGll3sNqjSVLgp1mcUzuMRCtenttXpg
q1sXDJ7FXkxQy7UvO8eMmMzPqkFm7KfLYZjkXrYo5ZhG+nfXqs3/HSuJ1fVe8m4/
+PVW1Uw2QtkfJiZVxOj60cV0lK/P1bJHah5xEtbnQ/flljitFiuo.JjTkoCNbeO
Sq6Kr0+LjJMaXmCeA6kZ7RXLHak/O5aR0BpXJCPUBIEFVnu6dftolO6JPcqMbW+
5oI4NHajunHz0eTgOuFBsV3EVjYmH7rabV832ikY0MARWRh/b/osUv+Ht9BIUHaY
pLoGPXkLeSsCDo714Z/dufLGUEFcNxx/QAmhWGiKH4MacMvKBVE+2uzcMAWqnyEW
Oaz0bI893YUtbQbt2rdQgVdHHXjWCmQ2YnMWv4pSBAwK7rPOrtehmdsmiOuit6x
FHvHPldc1o38u0Jg4d0LjCv/rRBdQwasJzr46dwJBOCv5rQ9Rkuul+6rhFQGns1G
```



```
jtJxAZIWwu8ZqD572a3jYVbIl/qBAW+dM3Fnt9NRqFUJVtdrd/AIAFm/OIwIVACF
Cbmjyxqliv/WYxNdJFL+IsHMX2Ury9TT2LMaDXxez6LRPMxJxRZSFhuyoYAqDYKT
NI1LSODDRZ1WYICOkPXlrSfzyig=
=IN1g
-----END PGP MESSAGE-----
```

以上为输出

这里，我们可以看出，加密之后生成mypgptest.asc文件，其中的内容已经经过加密了。

3，公钥解密过程

1)使用如下命令进行解密：

```
[root@lv-k gpg_test]# gpg -o mydecrypt --decrypt mypgptest.asc
```

以下为输出

您需要输入密码，才能解开这个用户的私钥：“quietheart (test)”

2048 位的 ELG-E 密钥，钥匙号 2BBE2C91，建立于 2011-06-14 (主钥匙号 DDBA2DEA)

请输入密码：<=====输入密码

您需要输入密码，才能解开这个用户的私钥：“quietheart (test)”

2048 位的 ELG-E 密钥，钥匙号 2BBE2C91，建立于 2011-06-14 (主钥匙号 DDBA2DEA)

gpg: 由 2048 位的 ELG-E 密钥加密，钥匙号为 2BBE2C91、生成于 2011-06-14

“quietheart (test)”

以上为输出

这里，使用-o指定输出文件，使用--decrypt指定待解密文件。我们可以看出，解密的时候，我们需要输入密码，才能解密成功，而密码就是之前我们创建钥匙对时候输入的那个密码。注意，这里因为生成密钥，加密，解密都在一个机器上进行，所以可以成功解密，如果把加密之后的文件拿到别的机器上面，就无法解密了，如果要在其它机器上面解密，我们需要把本地私钥导出，发送给待解密的机器，然后在解密的机器上把刚刚导出的私钥导入，就行了。后面会说到如何解密。如果想要别人和你使用这个加密的方法通信，需要把你的公钥导出，发给别人，然后他们把这个公钥导入，在使用前面加密的方法用这个公钥加密数据并且发送给你，你再用你自己的私钥解密，得到解密后的原始数据,这也是公钥加密通信使用的常用方法。后面会详细讲述如何导出本地的公钥和私钥，以及如何在其它机器上面导入之前导出的密钥。

2)查看解密生成的文件

```
[root@lv-k gpg_test]# ls
```

```
mydecrypt mypgptest mypgptest.asc
```

```
[root@lv-k gpg_test]# cat mydecrypt
```

```
hello!
```

```
welcome come to here
```

```
today is 2011-06-14
```

```
[root@lv-k gpg_test]#
```

***导出（备份）公钥：**

```
[root@lv-k gpg_test]# gpg -o mypubkey --export DDBA2DEA
```

这里使用格式“gpg -o keyfilename --export KeyID”，使用-o指定生成的导出文件名称，使用--export指定想要导出的密钥ID，如果没有KeyID则是备份所有的公钥，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。导出的公钥，可以发布，其它人只要导入你发布的公钥，就能用这个公钥加密数据并且发送给你，你再用你自己的私钥解密，得到解密后的原始数据,这也是公钥加密通信使用的常用方

法。

*以文本方式导出公钥：

```
[root@lv-k gpg_test]# gpg -a -o mypubkeyascii --export DDBA2DEA
```

导出的文件mypubkeyascii可以查看其内容，这里内容如下：

```
mypubkeyascii
```

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

```
Version: GnuPG v1.4.5 (GNU/Linux)
```

```
mQGiBE320yYRBACUvTPS5Jxv2xamVudHL//PBhJESyUzHQcGtb/CPzyAkOaNvQ7U
V0DEjd+m61SAv7wwWltn9D3fOjZ09EdGY/9mHeVEOGLPdB7Seo28UyRtr6vHCrrl
1qHBS6I6jQ/iATDg+07O9hgDp5eCebI4aNyRGeRARx0t5vKgulF+FEzmlwCghWM8
toTSnP/bC1VePRXEZ9Uw+OkD/0cYIH2AZIKdbtjQ9J6F7AtgiPqRnjiTL7mOj6Xa
ncjThX9XGH5DUMoqR6Gaq9/eDtlefMwHCweiqdm1TNnvU/b7qDpw1TjxVLkHHVZk
8F4f8LsSkJNuLiqOwXwOJkuevQZ+Y3quF3nsOtURSx5nuKkekp4toOWSe3fEFWd
BQN5A/9Cxi55KWXzzYD8v+z0xiC3HvIWgUp0GTfcjSOfdhs9xtKe33LuGy7hoEt0
TfQpTnz8llcbHHYvtmAaaGrwBnBxoYrLrH5WrHx1n34ZdOszYK5p/yKW+g+0eLt
dv/f4uXEhfPInOk21X5PRGFcyNo3fMVGIHo1S8oiEX4fuEvvWrQqcXVpZXRozWFy
dCAodGVzdCkgPHF1aWV0X2hlYXJ0MDAwQDEyNi5jb20+iGAEEExECACAFak320yYC
GwMGCwkIBwMCBBUCCAMEFgIDAQIeAQIXgAAKCRDbssC3bot6o2bAJ918FqgoPMs
4eIQOWBDVs5sS9hSbgCfTbOggNKPJtCPJq4zr/jaPctE7Qi5Ag0ETfbTLBAIAKUn
cibGAlBwN5nW2NCN52oXWAsIXTDnUP7FmKi8K6B2PO7Bex2e95f44iOyBtLo73KF
vH64Wbl7wGQ0IOmn/AQbivoa9oaBi4+2mf9PbGMvo2Nrii8xcvBfvrh8Q7XsRoOH
```

```
obKv3inOGvxb0fkDUnlWkyYGtWCil5aK8b1jUACuB6Sbdtkeg3DrQp3BZjxgkVa
etgPnj70AtJfTFmn5WSIkBGJq2pbs7E9wNQC7Nvle62DHJj1nsPJg/d9ZdFWnQiW
HO6ik3bjRBLR4Iy82cjVOI2JmiaKVIPktpH9id5F/a3BlcaOF5wSovE4ssHdZGN
cvk4QPLr0X9G43ZBTmcAAwUH/07bUjqWu61ulq5XryRK99pTNLlwd1AdsB/HTgFp
UJxSZ7MRbtozvdBPVKFehAzZ13YSjiES0flL3oGBsEeEhCPyHY+N80R+zuLMeZTx
6kFNLlm/PEFNrnFpqGPEJZ/1HRyt+hTsfCFM4nYzhul7s10XKGiHHN1wUMc/ixt
+tmk/dD6IA/QnhI1WUMzF2XgJ3Q4ZmLCSNV7CXPEt044J8ZnuTT+HwO4AX7bfSt
PTpyi7LMD/BVfOikczvf38Bz/IUnXi8x1pjBglZBE7jvkqc+nqe7BYQ/ga27c374
zQDIaMAXcZ0TR35T+ZiyavQHKiRSuAG0q3T2oO74yvb8AgmlSQQYEQIACQUCTfbT
LAIbDAAKCRDbssCl3bot6uJ8AJ4h8ShsY6DLjmJjitl+8iHnh5tZQACeOqHsQXjr
dei6gV8vMB87xwbWe5E=
=ynen
-----END PGP PUBLIC KEY BLOCK-----
```

一般，许多地方网上通过这个文本方式发布公钥。

***导出（备份）私钥：**

```
[root@lv-k gpg_test]# gpg -o mysubkey --export-secret-keys 2BBE2C91
```

如果没KeyID则是备份所有的私钥，-o表示输出到文件mysubkey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。

***导入私钥：**

```
gpg --import mysubkey
```

输入之后，输出如下：

```
#####以下为输出#####
```

```
gpg: 密钥 DDBA2DEA：私钥已导入
```

```
gpg: /home/lv-k/.gnupg/trustdb.gpg：建立了信任度数据库
```

```
gpg: 密钥 DDBA2DEA：公钥“quietheart (test)”已导入
```

```
gpg: 合计被处理的数量：1
```

```
gpg:      已导入：1
```

```
gpg:      读取的私钥：1
```

```
gpg:      导入的私钥：1
```

```
#####以上为输出#####
```

这里，如果导入公钥命令是一样的，不过指定的文件应该是"mypubkey"了。这个命令是另外一台机器上运行的，导入私钥之后那个机器就可以使用这个私钥解密数据了。一般来说我们都是发布公钥让人导入，而不是导入私钥匙。

实践发现，

*导入私钥之后，另外一台机器直接可以用对应的公钥加密，而不用导入公钥;这时候另外的那个机器也可以导出公钥，不过有一行内容和原始机器公共钥匙内容不一样，但是用这个公钥加密的数据也可以用原始的机器解密出来的。

*导入公钥之后，另外一台主机无法导出私钥，可以导出公钥，导出内容和原来一样。使用公钥加密之后，无法解密（因为没有私钥）。

无论导入的是公钥还是私钥，导入之后可以通过`gpg --list-keys`来查看导入的结果，而且从结果可以看到没有导入的那个配对(或者私钥或者公钥)的KeyID。

****签名与验证**

签名作用是验证明文、加密文件、密钥是来自正确的发送者的，没经过其它人的修改。签名使用的也是密钥对，与加密操作相同。只是在结果上，点不同。它只是在文件最后添上加密的验证信息（签名）。一旦文件有所改变，签名验证就会出错。比如我们ubuntu安装软件时，首先要用事先保存的密钥（大多从 <http://keyserver.ubuntu.com> 获得）验证软件源的签名，以保证我们连的是正确的安全的下载服务器。

*查看之前的目录和文件如下：

```
[root@lv-k gpgtest]# pwd
```

```
/root/tmpTrans/gpgtest
```

```
[root@lv-k gpgtest]# ls
```

```
mydecrypt
```

```
[root@lv-k gpgtest]# cat mydecrypt
```

```
hello!
```

```
welcome come to here
```

```
today is 2011-06-14
```

*生成签名，过程如下：

```
[root@lv-k gpgtest]# gpg -o mydecrypt.sig -s mydecrypt
```

```
#####以下为输出#####
```

```
您需要输入密码，才能解开这个用户的私钥：“quietheart (test)”
```

```
1024 位的 DSA 密钥，钥匙号 DDBA2DEA，建立于 2011-06-14
```

```
请输入密码：<====这里输入你的密码
```

```
#####以上为输出#####
```

```
[root@lv-k gpgtest]# ls
```

```
mydecrypt mydecrypt.sig
```

这里可以看到生成了mydecrypt.sig文件(其内容是乱码)，其中，mydecrypt是原文件，mydecrypt.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私钥密码。

**产生文本格式的签名*

```
[root@lv-k gpgtest]# gpg -o mydecrypt.sig --clearsign mydecrypt
```

```
##### 以下为输出#####
```

您需要输入密码，才能解开这个用户的私钥：“quietheart (test)”

1024 位的 DSA 密钥，钥匙号 DDBA2DEA，建立于 2011-06-14

请输入密码：<====这里输入你的密码

```
##### 以上为输出#####
```

```
[root@lv-k gpgtest]# cat mydecrypt.sig
```

```
-----BEGIN PGP SIGNED MESSAGE-----
```

```
Hash: SHA1
```

```
hello!
```

```
welcome come to here
```

```
today is 2011-06-14
```

```
-----BEGIN PGP SIGNATURE-----
```

```
Version: GnuPG v1.4.5 (GNU/Linux)
```



```
iD8DBQFOCFkQ27LApd26LeoRAvC+AJ9xp1HLT6zup7AZtan5qpQrpQyn1QCfV0Zy
HG+z+/hxfibs9pzo6ODYDG4=
=o1+k
-----END PGP SIGNATURE-----
```

这里，通过以上可知产生的mydecrypt.sig同样包含原文件和签名，文件是文本格式的，原文件不变。解开和验证签名的方法

***验证签名：**

```
[root@lv-k tmp]# gpg --verify mydecrypt.sig
```

gpg: 于 2011年06月27日 星期一 17时58分39秒 CST 创建的签名，使用 DSA，钥匙号 DDBA2DEA

gpg: 完好的签名，来自于“quietheart (test)”

这里，在验证之前，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参数(后面会讲到)。

***将签名文件恢复**

```
[root@lv-k tmp]# gpg -o my --decrypt mydecrypt.sig
```

gpg: 于 2011年06月27日 星期一 17时58分39秒 CST 创建的签名，使用 DSA，钥匙号 DDBA2DEA

gpg: 完好的签名，来自于“quietheart (test)”

```
[root@lv-k tmp]# cat my
```

hello!

welcome come to here

today is 2011-06-14

这里，不需要输入密码，生成的文件my和原来的文件名称一样。

*签名并加密：

```
[root@lv-k gpgtest]# gpg -o mydecrypt.sig -ser quietheart mydecrypt
```

以下为输出

您需要输入密码，才能解开这个用户的私钥：“quietheart (test)”

1024 位的 DSA 密钥，钥匙号 DDBA2DEA，建立于 2011-06-14

请输入密码：<====这里输入你的密码

以上为输出

这里，无法直接通过"gp -v mydecrypt.sig"对文件mydecrypt.sig进行验证。而是在解密恢复文件的时候直接验证了，后面会说到。

*恢复加密的签名文件：

```
[root@lv-k gpgtest]# gp -o my --decrypt mydecrypt.sig
```

以下为输出

您需要输入密码，才能解开这个用户的私钥：“quietheart (test)”

2048 位的 ELG-E 密钥，钥匙号 2BBE2C91，建立于 2011-06-14 (主钥匙号 DDBA2DEA)

请输入密码：<====这里输入你的密码，输入之后提示自动消失

gp: 由 2048 位的 ELG-E 密钥加密，钥匙号为 2BBE2C91、生成于 2011-06-14

“quietheart (test)”

gp: 于 2011年06月27日 星期一 18时11分27秒 CST 创建的签名，使用 DSA，钥匙号 DDBA2DEA

gpg: 完好的签名，来自于“quietheart (test)”

#####以上为输出#####

```
[root@lv-k gpgtest]# cat my
```

hello!

welcome come to here

today is 2011-06-14

这里，解密之后进行验证，而不是直接验证，因为不能通过"pgp --verify mydecrypt.sig"直接验证加密的签名文件。

***分离式签名：**

```
[root@lv-k gpgtest]# gpg -o mydecrypt.sig -ab mydecrypt
```

#####以下为输出#####

您需要输入密码，才能解开这个用户的私钥：“quietheart (test)”

1024 位的 DSA 密钥，钥匙号 DDBA2DEA，建立于 2011-06-14

请输入密码：<====这里输入你的密码，输入之后提示自动消失

#####以上为输出#####

```
[root@lv-k gpgtest]# cat mydecrypt.sig
```

-----BEGIN PGP SIGNATURE-----

Version: GnuPG v1.4.5 (GNU/Linux)

```
iD8DBQBOCZDP27LApd26LeoRArTQAJ9q13/4jVvJbg5f83lNnoC1Gq111wCfWRm8
1awHtUl2sN9SWNt0qNoFQHw=
=ySnh
-----END PGP SIGNATURE-----
```

这里，mydecrypt.sig仅包含签名，分离式签名的意思是原文件和签名是分开的。b选项表示分离式签名 detach-sign.

***对分离的签名进行验证：**

```
[root@lv-k gpgtest]# gpg --verify mydecrypt.sig mydecrypt
```

gpg: 于 2011年06月28日 星期二 16时29分03秒 CST 创建的签名，使用 DSA，钥匙号 DDBA2DEA

gpg: 完好的签名，来自于“quietheart (test)”

这里，和前面的验证方式不同，因为签名和数据文件是分离的，所以验证时，指明签名文件"mydecrypt.sig"的同时也要指明相应的数据文件"mydecrypt"。因为签名是分离的，所以不需要使用"--decrypt"进行恢复，如果恢复那么也仅仅是打印出签名的信息（这里"恢复"的时候不用指明数据文件，会提醒你指出数据文件的位置）。

[其他]

****编辑公钥**

可以对公钥进行编辑，这里没具体实践，大致过程如下：

```
#gpg --edit-key someone
```

someone是别人的用户id,输入之后，出现命令提示符号。

>fpr <===输入这个表示查看someone的指纹，核对信息真实性，这样之后签署。

>sign <===输入这个签署公钥，这样以后再使用它加密的时候不会产生警告了。

>check <===输入这个，检查someone已有的钥匙的签名。

>quit <===输入这个，退出交互，可能会提示你保存之前的设置。

**

参考：

<http://blog.chinaitlab.com/html/57/340757-64526.html>

http://blog.sina.com.cn/s/blog_44abafb201008zjo.html

<http://wiki.ubuntu.org.cn/GPG/PGP>

作者：**QuietHeart**

Email：**quiet_heart000@126.com**

日期：**2011年6月28日**

阅读(9149) | 评论(1) | 转发(5) |

上一篇：[Linux命令学习手册-ifconfig命令](#)

下一篇：[Samba服务配置和使用](#)


相关热门文章

 linux 常见服务端口

 xmanager 2.0 for linux配置


 【ROOTFS搭建】busybox的httpd...

 openwrt中luci学习笔记


 什么是shell

 linux dhcp peizhi roc

 关于Unix文件的软链接

 求教这个命令什么意思，我是新...

 sed -e "/grep/d" 是什么意思...

 谁能够帮我解决LINUX 2.6 10...

给主人留下些什么吧！~~



VIP_fuck

2015-12-15 11:21:50

NB，这个博文好。

[回复](#) | [举报](#)

评论热议

请登录后评论。

[登录](#) [注册](#)

- | | | | |
|------------------------------|-------------------------|---------------------------|----------------------------|
| 1 python学习手册 | 4 学习手册 | 7 免费网校 | 10 linux教程 |
| 2 学习计划 | 5 学习技巧 | 8 linux入门 | 11 学习 |
| 3 学生学习手册 | 6 学习的名言 | 9 学习手册 英文 | 12 学习方法 |



百度公益频道全新上线

[关于我们](#) | [关于IT168](#) | [联系方式](#) | [广告合作](#) | [法律声明](#) | [免费注册](#)

Copyright 2001-2010 ChinaUnix.net All Rights Reserved 北京皓辰网域网络信息技术有限公司. 版权所有

感谢所有关心和支持过ChinaUnix的朋友们

京ICP证041476号 京ICP证060528号