

密钥、证书生成和管理总结

目录

- 1. [openSSH的ssh-keygen工具](#)
- 2. [openSSL](#)
- 3. [keytool](#)
- 4. [GPG](#)
- 5. [PGP](#)
- 6. [Putty](#)
- 7. [其他未整理内容](#)

一、OpenSSH和ssh-keygen

OpenSSH是SSH协议的开源版本（SSH：Secure SHell）。使用SSH透过计算机网络实现加密通讯，可以进行远程控制，在计算机之间传送文件等等。SSH传输的数据都进行了加密，比telnet,rcp,ftp,rlogin,rsh等以明文传输密码的工具更安全。

OpenSSH提供了实现SSH协议的很多工具。其中ssh-keygen用于生成，管理和转换用于认证的密钥和证书。

ssh-keygen

公告

01697
03072

since 2015.11.20

昵称：XRacoon

园龄：2年9个月

粉丝：0

关注：5

+加关注

<	2016年3月						>
日	一	二	三	四	五	六	
28	29	1	2	3	4	5	
6	7	8	9	10	11	12	
13	14	15	16	17	18	19	
20	21	22	23	24	25	26	
27	28	29	30	31	1	2	
3	4	5	6	7	8	9	

搜索

找找看

谷歌搜索

- 常用链接
- 我的随笔
 - 我的评论
 - 我的参与

- **-b <bits>** 密钥长度，默认为2048
- **-t <type>** 密钥类型, rsa|des... (默认类型为rsa)
 - SSH1: RSA
 - SSH2: RSA, DSA, ECDSA
- **-N <password>** 新密码
- **-f <file>** 指定密钥文件，创建时会同时生成一个.pub结尾的公钥文件。
- **-C <comment>**
- **-c** 修改公钥或私钥文件中的注释
- **-p** 修改私钥文件密码
- **-P <password>** 旧密码
- **-e** 导出为其它格式的密钥文件，可以转换密钥类型
- **-i** 从其他格式的密钥文件导入，可以转换密钥类型
- **-m <PEM|PKCS8|RFC4716>** 与-e,-i配合使用，指明导出或导入的密钥文件格式
- **-y** 读入密钥并显示公钥

二、OpenSSL

OpenSSL 是一个强大的安全套接字层密码库，包括了加密算法，常用密钥和证书管理，SSL协议等功能。OpenSSL提供的命令非常多，这里只简单列出OpenSSL生成密钥和证书的一些操作（Window需要以管理员身份运行cmd）。

v1.0.1+密钥默认采用PKCS#8格式（之前版本为PEM）。

[查看openssl版本](#)

[最新评论](#)

[我的标签](#)

[我的标签](#)

[java\(8\)](#)

[Maven\(7\)](#)

[Git\(6\)](#)

[android\(5\)](#)

[SCM\(4\)](#)

[JSP\(3\)](#)

[linux\(2\)](#)

[Jenkins\(2\)](#)

[MVC\(2\)](#)

[MyBatis\(2\)](#)

[更多](#)

[随笔分类](#)

[.Net/C#\(4\)](#)

[Algorithm\(1\)](#)

[Android\(3\)](#)

[Ant\(2\)](#)

[Appium](#)

[Apple/Mac OS/iOS\(5\)](#)

[Database\(2\)](#)

[Docker](#)

[eclipse\(1\)](#)

[Gradle](#)

[Java\(17\)](#)

[Jenkins\(2\)](#)

[JVM](#)

[Maven\(7\)](#)

[Network](#)

[RESTful\(1\)](#)

[Robot Framework](#)

[Ruby](#)

[SCM\(4\)](#)

```
openssl version
openssl version -a
```

1. 密钥生成 (**genrsa**, **genpkey**, **req**在证书请求时同时生成密钥, **gendh**, **gendsa**)

(1) openssl genrsa [options] [bits_num]

- **-out <file>** 指定输出文件，不指定时在终端显示密钥内容
- **-passout pass:<password>** 设置私钥文件密码
- **-f4** 使用F4(0x10001)作为公钥的E参数（默认）
- **-3** 使用3作为公钥的E参数
- **-des**, **-des3**, **-aes128**, **-aes192**, **-aes256** 指定加密算法（默认不加密）

默认生产的密钥格式为PEM。**openssl**默认只生成了私钥文件，当需要提取公钥时使用**rsa**命令。

示例



#生成RSA密钥对。位长度为2048，保持到rsakey0.pem文件中。

```
openssl genrsa -out rsakey0.pem 2048
```

#生成RSA密钥对。使用DES3加密，密钥使用密码保护，位长度为1024

```
openssl genrsa -des3 -out rootca.key -passout pass:123456 1024
```



Security(2)

Server

Unix/Linux/BSD(2)

Web前端(1)

Windows(1)

测试

机器学习

架构/设计模式

其他语言

云/大数据/数据挖掘

随笔档案

2016年3月 (2)

2016年2月 (6)

2016年1月 (2)

2015年12月 (4)

2015年11月 (11)

2015年10月 (5)

2015年9月 (11)

2015年8月 (10)

2015年7月 (1)

2015年6月 (1)

2015年5月 (3)

最新评论

1. Re:《Java核心技术卷一》笔记 多线程和同步
(高层实现)

在创业中，已经拿到投资，游戏产品已经完成，
现在在处理游戏细节，等细节处理完成后做上线
钱的准备。

现在在招人，请问有兴趣加入么，我的QQ55763

--mgtec7758

阅读排行榜

1. Android shell 命令总结(218)

(2) openssl genpkey [options] v1.0.1+

示例

```
#生成RSA密钥，位长度为2048，格式为DER
openssl genpkey -algorithm RSA -out rsapriKey.pem -pkeyopt
rsa_keygen_bits:2048 -outform DER
```

(3) openssl req请求时生成新的密钥对

```
openssl req -x509 -days 365 -newkey rsa:2048 -keyout private.pem -
out public.pem -nodes
```

2. 密钥文件管理和转换 (rsa, pkey)

(1) openssl rsa [options] <infile> >outfile

- **-in <infile>** 输入密钥文件
- **-passin pass:<password>** 输入密钥保护密码
- **-inform <DER|NET|PEM>** 输入密钥格式，默认为PEM
- **-out <outfile>** 输出密钥文件
- **-outform <DER|NET|PEM>** 输出密钥格式，默认为PEM
- **-passout pass:<password>** 输出密钥保护密码
- **-pubin** 指示输入的是公钥，默认输出的是密钥对或私钥
- **-pubout** 输出公钥到文件（公钥一般无需加密）
- **-des, -des3, -aes128, -aes192, -aes256** 指定密钥加密算法
- **-text** 明文输出密钥参数
- **-noout** 不输出密钥到文件

2. 密钥、证书生成和管理总结(190)
3. MyBatis使用总结(171)
4. Android adb 命令使用总结(165)
5. maven-surefire-plugin总结(152)

评论排行榜

1. 《Java核心技术卷一》笔记 多线程和同步（高层实现）(1)

- -check 验证一致性
- -modulus 显示RSA密钥模值

示例



#提取密钥公钥到单独的文件

```
openssl rsa -in rsakey0.pem -pubout -out rsakey0.pub
```

#转换密钥格式 (DER->PEM)

```
openssl rsa -in rsakeypair.der -inform DER -out rsakeypair.pem
```

#改加密算法，移除密码保护

```
openssl rsa -in rsakeypair.pem -passin pass:123456 -des3 -out  
rsakeypair1.pem
```



(2) openssl pkey [options] v1.0.1+

(3) 将PEM格式密钥转换成Java JCE 能使用的DER格式密钥的另一种方式

```
openssl pkcs8 -topk8 -inform PEM -outform DER -in <rsa_pem.key> -out  
<pkcs8_der.key> -nocrypt
```

(4) OpenSSL公钥和OpenSSH公钥格式转换

OpenSSL生成的公钥格式和OpenSSH公钥格式不一致，把OpenSSL生成的公钥用于配置SSH连接，验证会失败。

OpenSSL公钥（PEM）格式为：



```
-----BEGIN PUBLIC KEY-----  
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC7vbqajDw4o6gJy8UtmIbkcpnk  
O3Kwc4qsEnSZp/TR+fQi62F79RHWmwKotFmwteURgLbj7D/WGuNLGOfa/2vse3G2  
eHnH15CB8ruRX9fBl/KgwCVr2JaEuUm66bBQeP5XeBotdR4cvX38uPYivCDdPjJ1  
QWPdspTBKcxeFbccDwIDAQAB  
-----END PUBLIC KEY-----
```



OpenSSH公钥格式为：



```
ssh-rsa  
AAAAB3NzaC1yc2EAAAADAQABAAQgQC7vbqajDw4o6gJy8UtmIbkcpnkO3Kwc4qsEnSZp  
/TR+fQi62F79RHWmwKotFmwteURgLbj7D/WGuNLGOfa/2vse3G2eHnH15CB8ruRX9fBl/  
KgwCVr2JaEuUm66bBQeP5XeBotdR4cvX38uPYivCDdPjJ1QWPdspTBKcxeFbccDw==
```



- 从私钥重新生成**OpenSSH**格式公钥

```
ssh-keygen -y -f priKey.pem > sshPubkey.pub
```

- 将**OpenSSL**格式公钥转换成**OpenSSH**格式

```
ssh-keygen -i -m PKCS8 -f sslPubKey.pub > sshPubKey.pub    #-m支持  
PEM，PKCS8，RFC4716
```

- 将 **OpenSSH** 格式公钥转换成 **OpenSSL** 格式公钥

```
ssh-keygen -e -m PEM -f sslPubKey.pub >sshPubKey.pub        #-m支持  
PEM，PKCS8，RFC4716
```

3.使用密钥（rsautl）

openssl rsautl [options]

- -in <file> 输入需加密解密文件
- -out <file> 输出加密或解密后的文件
- -inkey <file> 输入密钥文件
- -passin pass:<password> 输入密钥文件的保护密码
- -keyform <PEM|DER|NET> 输入密钥文件格式，默认为PEM
- -pubin 指明输入的是公钥文件
- -certin 指明输入的是包含公钥的证书
- -sign 使用私钥签名签名（加密）
- -verify 似乎用公钥验证签名（解密）
- -encrypt 使用公钥加密
- -decrypt 使用私钥解密
- -hexdump 输出十六进制
- -pkcs, -ssl, -raw, -oaep 数据补齐方式，默认为pkcs

使用RSA加密时，要求被加密的数据长度与RSA密钥长度一致，数据过短时会补齐。过长时会分段加密。实际应用中很少会使用非对称加密算法对大的文件进行加密操作，而是使用对称加密算法加密文件，然后再使用<非对称加密算法>对<对

称算法的密钥>进行加密。

示例：



#使用公钥加密

```
openssl rsautl -in test.txt -out test_enc.txt -inkey rsakeypair.pub  
-pubin -encrypt
```

#使用私钥解密

```
openssl rsautl -in test_enc.txt -out text_dec.txt -inkey  
rsakeypair.pem -decrypt
```

#使用私钥签名

```
openssl rsautl -in test.txt -out test_sign.txt -inkey rsakeypair.pem  
-sign
```

#使用公钥验证签名

```
openssl rsautl -in test_sign.txt -out test_unsign.txt -inkey  
rsakeypairi.pub -pubin -verify
```



4. 证书管理

- 生成**X509**格式的自签名证书

```
openssl req -x509 -new -days 365 -key rsakey.pem -out cert0.crt
```


会要求输入区别名DN的各项信息（国家，城市，组织，姓名，email等）。

根证书 是认证中心机构（Certificate Authority）给自己签发的证书，签发者就是自身，是信任链的起点。里面包含了CA信息、CA公钥、用自身的私钥对这些信息的签名。下载并使用根证书就表示你信任它的来源机构，自然也信任证书以下签发的所有证书。某个证书可以用签发他的证书中的公钥验证，签发他的证书又需要上一层签发证书来验证，直到通过根证书中的公钥验证，那么这个证书就是可信任的。

- 生成要求根证书签发子证书的请求文件

```
openssl req -new -key rsakey1.pem -out subcertreq.csr
```

会要求输入区别名DN的各项信息（国家，城市，组织，姓名，email等），还需要额外属性：密码 和 可选公司名。

- 使用根证书签发子证书

```
openssl x509 -req -in subcertreq.csr -CA cert0.crt -CAkey rsakey0.pem  
-CAcreateserial -days 365 -out subcert.crt
```

也可创建一个ca的配置文件，通过ca管理子命令来签发子证书（未试验）

```
openssl ca -config ca.config -out user.crt -infiles user.csr
```

- 将证书和密钥打包为pkcs12格式的库中

```
openssl pkcs12 -export -in subcert.crt -inkey rsakey1.pem -out  
subcert.p12
```

需要输入pkcs12文件密码。

- 查看证书内容

```
openssl x509 -noout -text -in rootca.crt
```

- 验证证书

```
openssl verify -CAfile rootca.crt subcert.crt
```

用rootca.crt的公钥验证subcert.crt中的签名

从证书中提取公钥

```
openssl x509 -in cert.pem -noout -pubkey > pubkey.pem
```

提取密钥对

```
openssl pkcs12 -in cert.pfx -nocerts -nodes -out keypari.pem
```

从PKCS#8提取公钥

```
openssl req -in public.pem -noout -pubkey
```

查看**pkcs7**签名内容的证书信息

```
openssl pkcs7 -in <pkcs7signedFile> -inform DER -print_certs
```

三、Keytool工具管理证书

keytool是Java提供的密钥、证书和证书库管理工具。可以完成生成密钥，生成证书等各种操作。

keytool的子命令如下：

- | | |
|-----------------|-----------------|
| -certreq | 生成证书请求 |
| -changealias | 更改条目的别名 |
| -delete | 删除条目 |
| -exportcert | 导出证书 |
| -genkeypair | 生成密钥对 |
| -genseckey | 生成密钥 |
| -gencert | 根据证书请求生成证书 |
| -importcert | 导入证书或证书链 |
| -importkeystore | 从其他密钥库导入一个或所有条目 |
| -keypasswd | 更改条目的密钥口令 |
| -list | 列出密钥库中的条目 |
| -printcert | 打印证书内容 |
| -printcertreq | 打印证书请求的内容 |
| -printcrl | 打印 CRL 文件的内容 |
| -storepasswd | 更改密钥库的存储口令 |

还可以使用 **keytool -command_name -help** 查看各个子命令的帮助信息

- **pkcs12**库中证书导入**jks**证书库（**java keystore**格式）

```
keytool -importkeystore -srckeystore subcert.p12 -destkeystore  
subcert.jks -srcstoretype pkcs12
```

需要输入目标库的密码和源库的密码，如果**jks**库文件不存在的话会自动生成。

- 导入证书到**jks**库

```
keytool -importcert -keystore subcert.jks -alias rootca -file  
rootcert.crt
```

需要输入目标库密码和是否信任添加的证书。**-alias**可以省略，如果**jks**库文件不存在的话会自动生成。

证书库或者说密钥库中即可以存放密钥也可存放证书，如果只包含证书（证书中有公钥）而不包含私钥，这样生成的库就是**trust**库。

查看证书指纹

```
keytool -list -keystore <keystoreFile> -alias <aliasName>
```

等待补充.....

四、GPG

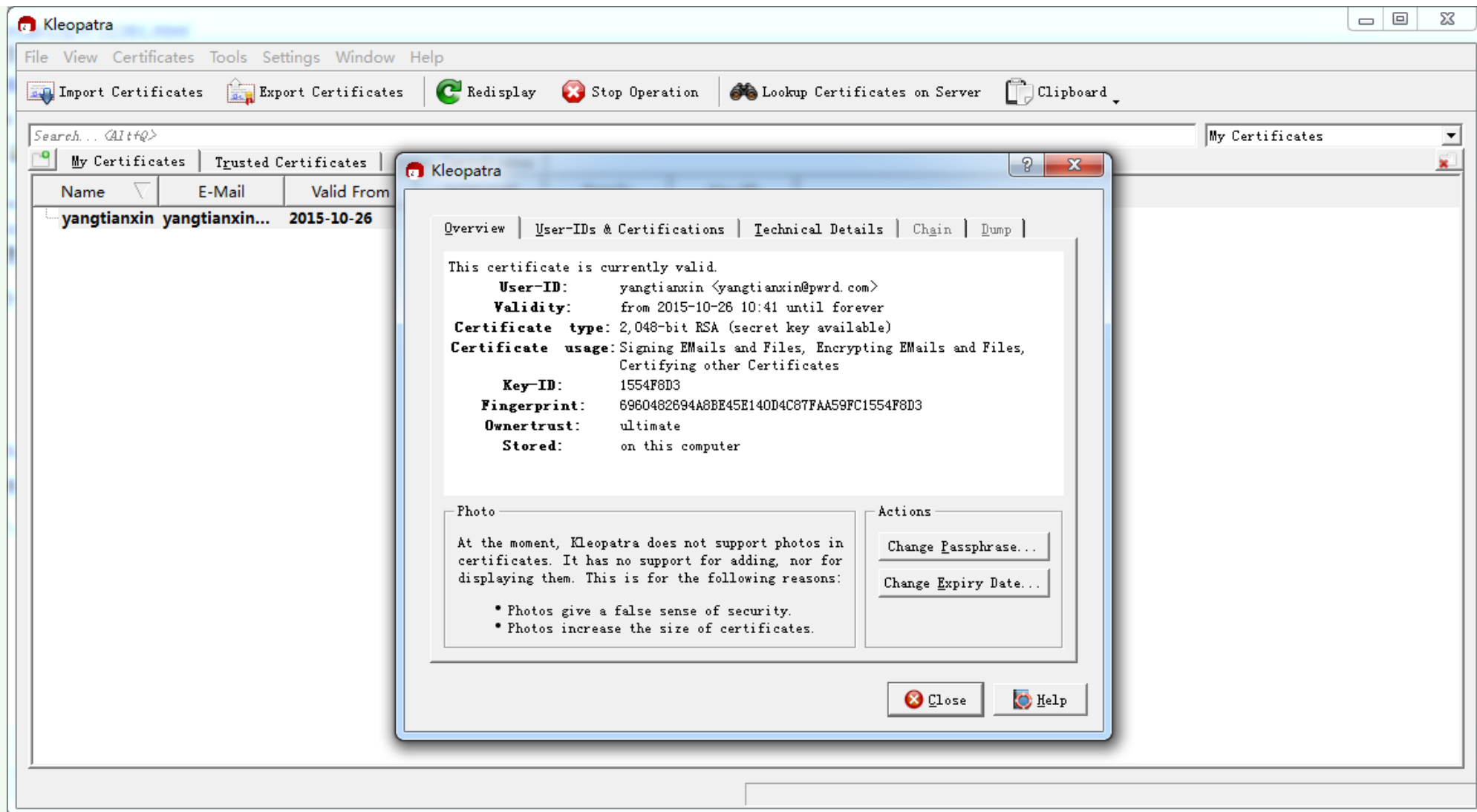
先整理下三个概念：

- **OpenPGP** 是一种加密标准和协议（RFC4880）。
- **PGP**（Pretty Good Privacy）是这个标准的闭源商业实现。
- **GPG**（GnuPG）是这个标准的开源实现。

可见PGP和GPG是基于同一个标准的两套独立的工具，都能处理加解密、签名等功能，GPG是开源免费的。

官方文档：<https://www.gnupg.org/documentation/manuals/gnupg/>

GPG使用~/.gnupg目录。GPG除了基本的GnuPG核心命令功能外，GPG提供了一个名为Kleopatra的GUI密钥和证书管理工具，使用起来非常方便。[Kleopatra的基本用法](#)。



GPG命令详解

-a 以ASCII输出

功能选择

`--sign | -s` 可以结合`--encrypt`或`--symmetric`或两者同时使用。使用默认的密钥。选择密钥使用`--local-user`和`--default-key`选项。

`--clearsign` 签名，内容可读。选择密钥使用`--local-user`和`--default-key`选项。

`--detach-sign | -b` 创建独立的签名

`--encrypt | -e` 非对称加密，可以结合`--sign`、`--symmetric`或两者同时使用

`-c | --symmetric` 对称加密。默认输出扩展名为`.gpg`，可以用`-o`指定输出文件，默认加密算法为`CAST5`，可以结合`--cipher-algo`指定算法。也可以配合`--sign`或`--encrypt`或两者同时使用。

等待补充.....

示例：(<http://blog.chinaunix.net/uid-9525959-id-2001824.html>)



```
gpg -c myfile [-o myfile.gpg]           #对称加密
gpg -d myfile.gpg -o myfile.decrypt      #对称解密
gpg --genkey                             #交互式创建密钥
gpg --list-keys                          #查看已有的密钥
gpg -e -a -r privkey file [-o descfile]  #加密（需要公钥）。 -e | --
encrypt, -a ASCII输出， -r 指定加密的用户ID。
gpg -o descfile --decrypt file.asc       #解密（需要私钥），会提示输
入密钥密码。
gpg -o pubkeyfile --export KeyID [-a]    #导出公钥（证书），如果未指
定KeyID，则备份所有公钥， -a ASCII输出否则为二进制格式。
gpg -o subkeyfile --export-secret-keys KeyID [-a] #导出私钥
gpg --import subkeyfile                  #导入私钥
gpg -o signedfile -s file                #签名（需要私钥）
```

```
gpg -o txtsignedfile --clearsign file #保护原文的签名
gpg --verify signedfile #验证签名（需要公钥）
gpg -o encryptsignedfile -s prikey file #签名并加密（恢复时验证，不能通过--verify直接验证）
gpg -o file --decrypt encryptsignedfile #恢复加密的签名文件并验证
gpg -o filesign -b -a file #-b分离的签名，生成文件只包含签名信息
gpg --verify filesign file #分离的签名验证
gpg --edit-key userID #编辑公钥，进入自命令环境：
fpr 查看指纹， sign 签署公钥（加密是不再产生警告），check 检查已有的钥匙的签名， quit 退出
```



五、PGP

PGP是OpenPGP标准的闭源商业版本。pgp提供了图形化工具和命令行工具，命令行工作主要有以下几个：

- pgpk.exe 密钥生成和管理
- pgpe.exe 加密
- ggps.exe 签名
- pgpv.exe 解密和校验
- pgpo.exe 模拟2.6.3老版本命令行方式

示例（<http://blog.chinaunix.net/uid-7673620-id-2598657.html>）



```
pgpk -g
产密钥
```

#交互式生

pgpk -x keyname [-o pubkeyfile]	#导出公钥
pgpk -a pubkeyfile	#将密钥加
入密钥环(pubring.pkr文件)	
pgpk -r keyname	#从密钥环
删除公钥	
pgpe -r keyname originfile [-o encrypedfile] [-a]	#加密, -a
表示生成ASCII文档而不是二进制	
pgpv encrypedfile -o decrypedfile	#解密(需
要密钥密码)	
pgps -u keyname originfile [-o signedfile] [-a]	#签名(需
要密钥密码), -a表示生成ASCII文档而不是二进制	
pgpv signedfile	#签名验证
并恢复原始文件	
pgp -ka KEYS	#导入密钥
pgp file.asc	#验证签名



六、PuTTY工具

PuTTY是一个免费小巧(主程序只有300多K)的Window/Unix平台的ssh,telnet客户端,附带xterm终端模拟器。功能丝毫不逊于商业的SecureCRT。官方网站: <http://www.chiark.greenend.org.uk/~sgtatham/putty/>。

PuTTY包含多个工具。

- PuTTY telnet和SSH客户端

- PSCP SCP客户端
- PSFTP SFTP客户端
- PuTTYtel telnet客户端
- Plink PuTTY后端的命令行接口
- Pageant SSH认证代理，(PuTTY、PSCP、PSFTP、Plink都会用到)
- PuTTYgen RSA和DSA密钥生成转换工具

PuTTYgen

本文主要内容是密钥的管理，所有只说明PuTTYgen的使用。

PuTTYgen使用的密钥存储格式类似于OpenSSL（密钥文件扩展名.ppk，其中同时包含公钥和私钥）。

公钥格式



```
----- BEGIN SSH2 PUBLIC KEY -----  
Comment: "imported-openssh-key"  
AAAAB3NzaC1yc2EAAAABIwAAAQEA5of83WbXgRVTUFsOI49wacuEp253YuzZW1Lk  
.....  
vw7Oc6o357ipDPgWk9c82s+7Asov4OgX5o6hg3R7418Gc0lkqw==  
----- END SSH2 PUBLIC KEY -----
```



私钥格式(.ppk)

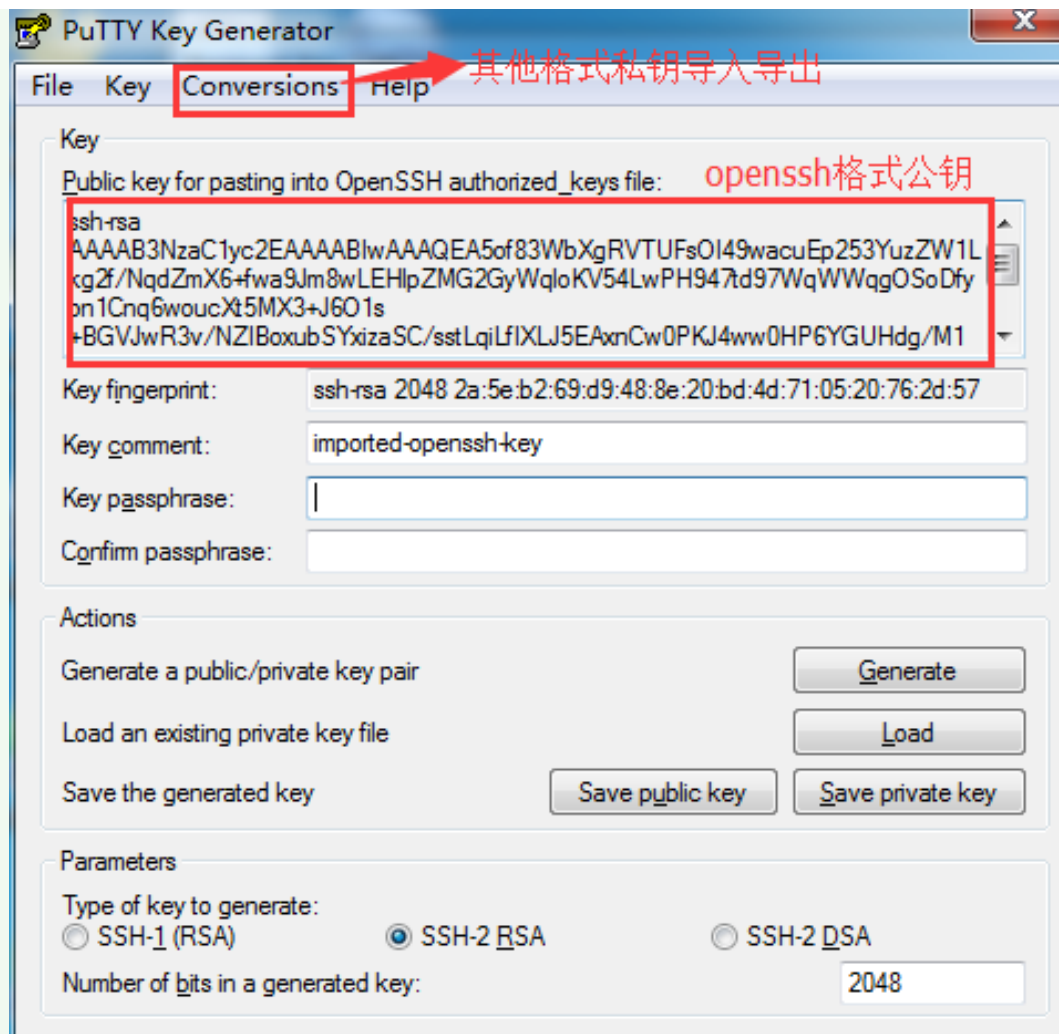


```
PuTTY-User-Key-File-2: ssh-rsa  
Encryption: aes256-cbc
```

```
Comment: imported-openssh-key
Public-Lines: 6
AAAAB3NzaC1yc2EAAAABIwAAAQEA5of83WbXgRVTUFsOI49wacuEp253YuzZW1Lk
...4行...
.....
vw7Oc6o357ipDPgWk9c82s+7Asov4OgX5o6hg3R7418Gc0lkqw==
Private-Lines: 14
xc2m3BR30d7GZG/QVatu8Xm5PfYYMXRfPobGtbqLsWWI9Wzfjs/InEVFPcG8kJ6v
...12行...
GANvUGQcgCyQTKVnqXmSMwPNZkqkJYZmRA43PBfiufVLV4JIitNxY8ol+enUfWwde
Private-MAC: 647a57a82038c05113fbd6a1758871b138a7a001
```



可以导入和导出OpenSSH和ssh.com格式的私钥（Conversions菜单下）。程序主界面中也会显示OpenSSH格式的公钥。



其他格式私钥导入导出

其他

OpenSSL生成的密钥文件(.pem)，签发请求文件(.csr)和证书文件(.cer)都是用纯文本格式保存了Base64编码后的信息(**PEM**格式)。可以用文本编辑器打开，内容如下：

```
-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEArlUX2v998Y+Ek/AzoDdsbW7ILyRpwXVBMDqmOf3ZZpbP/
4vo
.....省略若干行.....
buL0a0BOFi7dyJjWgtEyYTFQgYSeezHl/+XyOhuJITyvZWUpm5w=
-----END RSA PRIVATE KEY-----
```

首行和末尾行会被忽略，首行中显示了文件中存储的信息类型。

查看签名信息

```
jarsigner -verify -verbose -certs <jksSignedFile>
```

PEM格式 (PKCS#8)

pem是最常用的私钥和证书存储格式。其中通常会包含`--BEGIN XXXX--` and `--END XXXX--` 字符串，中间存放了Base64编码过的二进制数据。多个PEM证书或私钥可以包含在一个文件中，但最好还是将各个证书和私钥分开存放。扩展名可能为.pem, .crt, .cer, .key。

ASN格式

DER格式

主要用在Java，所有类型的证书和私钥都能用DER格式存放。文件中直接存储二进制数据，未进行Base64编码，无可读性的描述文字。扩展名可能为.cer, .der。

PVK格式（微软）

NET格式

P7B/PKCS#7文件（PEM）

包含“--BEGIN PKCS--” & “--END PKCS7--”字符串，只能存储证书和链证书，不能存放私钥。内容经过Base64编码，扩展名可能为.p7b，.p7c

PFX/PKCS#12文件（PEM）

可以加密的文件，可在一个文件中存放多个证书或密钥，主要用于证书和私钥的导入导出。扩展名一般为.pfx，.p12

PEM转DER

```
openssl x509 -outform der -in certificate.pem -out certificate.der
```

PEM格式证书转P7B

```
openssl crl2pkcs7 -nocrl -certfile certificate.cer -out certificate.p7b -  
certfileCAcert.cer
```

PEM格式转PFX

```
openssl pkcs12 -export -out certificate.pfx -inkey privateKey.key -  
in certificate.crt -certfile CAcert.crt
```

DER转PEM

```
openssl x509 -inform der -in certificate.cer -out certificate.pem
```

P7B转PEM

```
openssl pkcs7 -print_certs -in certificate.p7b -out certificate.cer
```

P7B转PFX（先转成PEM）

```
$ openssl pkcs7 -print_certs -in certificate.p7b -out certificate.cer  
$ openssl pkcs12 -export -in certificate.cer -inkey privateKey.key -  
outcertificate.pfx -certfile CAcert.cer
```

PFX转PEM（PFX中所有内容被存放到一个pem格式文件中）

```
openssl pkcs12 -in certificate.pfx -out certificate.cer -nodes
```

PKCS#8定义了私钥信息语法和加密私钥语法，X509定义证书规范，密钥通常使用DER和PEM进行编码存储，Java中JCE使用的是DER。openssl主要用的是PEM编码。PEM相对DER可读性更强，以BASE64编码，外围包上类似-----BEGIN RSA PRIVATE KEY-----。JCE没有对PEM直接支持，但是第三方那个包如bouncycastle可以解析。

很多SSH公钥使用openssh格式。其内容为ssh-rsa打头，RSA-1024结尾，中间是Base64编码：

```
ssh-rsa AAAAB3Nza.....cySYqQ== RSA-1024
```

参考资料：

OpenSSL文档：<https://www.openssl.org/docs/apps/openssl.html>

Keytool文

档：<http://docs.oracle.com/javase/8/docs/technotes/tools/unix/keytool.html>

<http://zhtx168.blog.163.com/blog/static/41601548200812503248/>

<http://blog.csdn.net/kimylrong/article/details/43525333>

<http://blog.csdn.net/jiftlixu/article/details/19836405>

<http://blog.csdn.net/zhymax/article/details/7683925>

http://www.360doc.com/content/13/1203/17/14797374_334193770.shtml

分类: [Security](#)


标签: [openssl](#), [加密](#), [证书](#), [keystore](#), [keytool](#), [PuTTYgen](#)

好文要顶

关注我

收藏该文







[XRacoon](#)
[关注 - 5](#)
[粉丝 - 0](#)
[+加关注](#)

0

推荐


0

反对

(请您对文章做出评价)

« 上一篇: [maven-surefire-plugin总结](#)
» 下一篇: [苹果电脑键盘符号记录](#)

posted @ 2015-08-11 21:23 XRacoon 阅读(191) 评论(0) 编辑 收藏
刷新评论 刷新页面 返回顶部

 注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问](#) 网站首页。

- 【推荐】50万行VC++源码：大型组态工控、电力仿真CAD与GIS源码库
- 【推荐】融云即时通讯云—专注为 App 开发者提供IM云服务
- 【推荐】UCloud开年大礼，充5000返1000；买云主机送CDN，详情点击

最新IT新闻：

- 拥有最大支付网络，VISA开放平台为数码支付找寻好应用
 - 苹果年度游戏《Downwell》 1人开发10个月
 - 柳传志、陈春花谈总裁的使命：如何在矛盾中前行？
 - 天天果园王伟：为风口而创业的人，早点回家洗洗睡吧
 - 期待已久的iOS 9.3升级，可能没你期待的那么好
- » 更多新闻...



最新知识库文章：

- 如何运维千台以上游戏云服务器
 - 架构漫谈（一）：什么是架构？
 - 架构的本质
 - 谷歌背后的数学
 - Medium开发团队谈架构设计
- » 更多知识库文章...

Copyright ©2016 XRacoon