《PKI 原理与技术》实验指导书

（学生用书）

课程名称：《PKI 原理与技术》

英文名称：PKI Principle and Technology

总学时：48　　学　分：2.5　　实验学时：　　上机学时：16

课程类别：专业领域课　　课程性质：选修

适用专业：信息安全

授课实验室：学院计算机网络安全实验室

一、实验（上机）教学目的与基本要求

1．实验目的

通过实验，基本掌握公钥基础设施（PKI）的有关理论、原理，基本掌握 PKI 的开发与使用相关技术。所以实验内容的设置也主要是结合上述的内容。

2．实验要求

（1）公钥证书的应用

（2）PKI 编程实践,利用 C 语言编写数字证书签发文档加密、签名等程序。

二、实验（上机）教学方式与考核方式

教学方式：老师先介绍各个实验的目的和主要内容，学生在学院计算机网络实验室分组进行。

考核方式：考核内容应包括实验过程的认真程度；实验记录、实验报告、实验课程总结记录书写情况；仪器设备操作使用情况；遵守实验室工作规章制度情况等。

成绩考核采用百分制或优秀、良好、中等、及格、不及格五级记分制，个别特殊课程无法按上述两种记分制评分的课程，可采用“合格”、“不合格”两级分制评定。学生考核、考试成绩 60 分以上或合格、及格取得该实验课程的学分。

三、实验（上机）参考书

1. 华南理工大学计算机学院“PKI 原理与技术”实验手册
2. 佘堃，郑方伟，《PKI 原理与技术》，电子科技大学出版社，2007.8

四、主要仪器设备

个人计算机。

实验一

一 、实验目的

学会签发根 CA 证书，使用根 CA 证书签发下级证书。

二 、实验要求

利用 OpenSSL 提供的命令行工具实现：

1. 生成根 CA 密钥对、生成自签名的根 CA 证书；
2. 生成普通个人用户的密钥对，并生成证书请求；
3. 以 CA 管理员的角色，给上一步生成的证书请求签发个人证书。

三 、实验环境

Linux 内核 2.6 及以上，安装有 OpenSSL。

四 、 实验步骤

1 、搭建实验环境。

此次实验需要的环境为：Linux+OpenSSL。

1) 首先查看 Linux 系统是否已经默认安装

[root@scut ~]# openssl version

如果在命令行中，可以输出 opensll 的版本信息，则说明系统已经默认安装好 OpenSSL，如果没有安装则可以参考以下步骤安装 OpenSSL

2） 安装 OpenSSL

a) 下载 OpenSSL 源代码包（www.openssl.org）

b) 解压 OpenSSL

tar –zxf opessl-xxx.gz

c) 安装 OpenSSL，注意可以把下列的路径换成自己的安装路径（其中 prefix 表示要把程序安装到哪个路径，opensslldir 指源代码所在的路径），如果要安装到默认的路径/usr/local/ssl，则可以直接输入./config。

[root@scut ~]#./config --prefix=/usr/local --opensslldir=/usr/local/openssl

[root@scut ~]# make

[root@scut ~]# make test

[root@scut ~]# make install

3）安装完成后，可以利用 1）中的命令查看安装的版本，至此环境搭建完成。

2 、 OpenSSL 建立自己的 CA

1) 环境准备

首先，需要准备一个目录放置 CA 文件，包括颁发的证书和 CRL(Certificate Revoke List)。这里我们选择目录 /var/MyCA。

然后我们在/var/MyCA 下建立两个目录，certs 用来保存我们的 CA 颁发的所有的证书的副本；private 用来保存 CA 证书的私钥匙。

除了生成钥匙，在我们的 CA 体系中还需要创建三个文件。第一个文件用来跟踪最后一次颁发的证书的序列号，我们把它命名为 serial，初始化为 01。第二个文件是一个排序数据库，用来跟踪已经颁发的证书。我们把它命名为 index.txt，文件内容为空。

```
[root@scut ~]# mkdir /var/MyCA
[root@scut ~]# cd /var/MyCA
[root@scut MyCA]# mkdir certs private
[root@scut MyCA]# chmod g-rwx,o-rwx private
[root@scut MyCA]# echo "01" > serial
[root@scut MyCA]# touch index.txt
```

第三个文件是 OpenSSL 的配置文件，创建起来要棘手点。示例如下（注意假如 ca 目录不是/var/MyCA 以下文件要作适当的修改，即把所有/var/MyCA 的路径换成你们自己的路径）：

```
[root@scut MyCA]#gedit openssl.cnf
```

文件内容如下：

```
[ ca ]
default_ca = myca

[ myca ]
dir = /var/MyCA
certificate = $dir/cacert.pem
database = $dir/index.txt
new_certs_dir = $dir/certs
private_key = $dir/private/cakey.pem
serial = $dir/serial

default_crl_days= 7
default_days = 365
default_md = md5

policy = myca_policy
x509_extensions = certificate_extensions

[ myca_policy ]
commonName = supplied
stateOrProvinceName = supplied
countryName = supplied
emailAddress = supplied
organizationName= supplied
organizationalUnitName = optional

[ certificate_extensions ]
basicConstraints= CA:false
```

我们需要告诉 OpenSSL 配置文件的路径，有两种方法可以达成目的：通过 config 命令选项；通过环境变量 OPENSSL_CONF。这里利用 config 选项。

2) 生成根证书 (Root Certificate)

我们需要一个证书来为自己颁发的证书签名，这个证书可从其他 CA 获取，或者是自签名的根证书。这里我们生成一个自签名的根证书。

首先我们需要往配置文件里面添加一些信息，如下所示，节名和命令行工具的命令 req 一样。我们把所有必要的信息都写进配置。

```
[ req ]
default_bits = 2048
default_keyfile = /var/MyCA/private/cakey.pem
default_md = md5
prompt = no
distinguished_name = root_ca_distinguished_name
x509_extensions = root_ca_extensions
[ root_ca_distinguished_name ]
commonName = My Test CA
stateOrProvinceName = HZ
countryName = CN
emailAddress = test@cert.com
organizationName = Root Certification Authority
[ root_ca_extensions ]
basicConstraints = CA:true
```

注意：在 OpenSSL 安装后会有一个 OpenSSL 配置文件，可以把那个文件 copy 过来，在上面修改。（查找文件命令：切换到/目录下然后输入 find -name openssl .cnf）

3) 万事俱备，我们可以生成根的密钥对和根证书了。

```
[root@scut MyCA]# openssl req -new -x509 -keyout ca.key -out ca.crt -config openssl.cnf
Generating a 2048 bit RSA private key
.............+++
.....................................................................+++
writing new private key to 'ca.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
```

以上命令执行过程中，会首先生成 CA 的密钥对，然后生成根证书。

3 、 生成普通个人用户的密钥对，并生成证书请求。

1) 生成普通个人用户的密钥对（key 文件）：

```
[root@scut MyCA]# openssl genrsa -des3 -out    client.key 1024
```

运行后，会出现很多提示信息，并要求我们输入相关的信息，我们按照要求输入就可以了。

```
Generating RSA private key, 1024 bit long modulus
.............++++++
.....................................................++++++
e is 65537 (0x10001)
Enter pass phrase for server.key:(asdfasdf)
Verifying - Enter pass phrase for server.key:(asdfasdf)
```

2) 生成普通个人用户的证书请求：

```
[root@scut MyCA]# openssl req -new -key client.key -out client.csr
```

运行命令后，我们需要按照要求输入相关信息：

```
Enter pass phrase for server.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
```

```
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [GB]:CN
State or Province Name (full name) [Berkshire]:GD
Locality Name (eg, city) [Newbury]:GZ
Organization Name (eg, company) [My Company Ltd]:SCUT
Organizational Unit Name (eg, section) []:TANGLAB
Common Name (eg, your name or your server's hostname) []:scut.tanglab.tang
Email Address []:12345@126.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:asdfasdf
An optional company name []:tanglab
```

至此，我们生成了普通用户的密钥对以及证书请求。

4 、 以 CA 管理员的角色给普通用户请求签发个人证书。

1) 给普通用户签发证书

```
[root@scut MyCA]# openssl ca -in client.csr -out client.crt -cert ca.crt -keyfile    ca.key -config
   openssl.cnf
Using configuration from openssl.cnf
Enter pass phrase for ca.key:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName                 :PRINTABLE:'CN'
stateOrProvinceName     :PRINTABLE:'GD'
localityName                :PRINTABLE:'GZ'
organizationName          :PRINTABLE:'SCUT'
organizationalUnitName:PRINTABLE:'TANGLAB'
commonName                    :PRINTABLE:'scut.tanglab.tang'
emailAddress                :IA5STRING:'12345@126.com'
Certificate is to be certified until Nov 10 01:52:28 2011 GMT (365 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

2) 用 OpenSSL 命令行，来验证我们生成的证书。

```
[root@scut MyCA]# openssl verify -CApath certs -CAfile ca.crt client.crt
client.crt: OK
```

实验二

一 、实验目的

掌握证书的结构，学会验证数字签名。

二 、实验要求

利用 OpenSSL 提供的库函数，用 C/C++编写：

1. 对于已经签发的一张个人数字证书，读出证书中每一个字段的内容，并显示出来，如果是二进制内容请用十六进制数来显示。
2. 验证证书中 CA 的数字签名是否为有效的（假设某根 CA 为可信的）。

三 、实验环境

Linux 内核 2.6 及以上，安装有 OpenSSL。

四 、实验步骤

1. 显示个人证书的内容。

默认情形下，OpenSSL 发布的证书是 Pem 格式的，而 OpenSSL 的库函数中，提供的了 X509 格式证书的查看函数，因此，我们首先需要下哦那个 Pem 格式的证书中，得到一个 X509 的数据结构：

```
b=BIO_new_file("client.crt","rb");
x=PEM_read_bio_X509(b,NULL,NULL,NULL);
BIO_free(b);
```

在获得 x509 的数据结构后，我们就可以调用 OpenSSL 的库函数把证书显示到标准输出：

```
b=BIO_new(BIO_s_file());
BIO_set_fp(b,stdout,BIO_NOCLOSE);
X509_print(b,x);
```

2. 验证证书中 CA 的数字签名是否为有效的

在程序中验证 CA 的签名是否有效，需要用到两个文件，一个是 CA 的证书，一个是待验证的证书，为此我们首先在程序中读取这个两个文件：

```
if (!(i = X509_LOOKUP_ctrl(lookup, X509_L_FILE_LOAD,
argv[1],(long)X509_FILETYPE_PEM, NULL)))
            app_abort("Can't open CAfile");
          ERR_clear_error();

          if ((in = BIO_new(BIO_s_file())) == NULL)
          app_abort("certfile BIO error");
          if (BIO_read_filename(in, argv[2])<=0)
     app_abort("open certfile error");
          if ((x = PEM_read_bio_X509(in, NULL, NULL, NULL)) == NULL)
     app_abort("load certfile error");
```

当读取这两个文件成功后，就可以调用 OpenSSL 所提供的库函数进行对 CA 签名的认证：

```
          if ((csc = X509_STORE_CTX_new()) == NULL)
               app_abort("ctx init error");
          X509_STORE_CTX_init(csc, cert_ctx, x, NULL);
          if ((i = X509_verify_cert(csc)) == 1)
          printf("Status=OK(%d)\n", i);
          else
          printf("Status=Error(%d)\n", i);
```

实验三

一 、实验目的

熟练掌握密钥对的生成以及利用密钥对做数字签名等操作。

二 、实验要求

利用 OpenSSL 提供的库函数，用 C/C++编写：

1. 编写程序生成 RSA 密钥对，并保存公钥到文件。
2. 对某个文件 Hash 后计算其数字签名，并把得到的签名信息保存到文件。
3. 利用公钥对所生成的数字签名进行验证。

三 、实验环境

Linux 内核 2.6 及以上，安装有 OpenSSL。

四 、实验步骤

1. 编写程序生成 RSA 密钥对。

在 OpenSSL 提供的库函数中，提供了生成 RSA 密钥对的函数，我们需要做的就是分配相应的存储密钥对的空间，然后调用函数生成密钥对即可。

```
RSA *rsa=RSA_new();
RSA *pub_key=RSA_new();
RSA *pri_key=RSA_new();
rsa=RSA_generate_key(bits,e,NULL,NULL);
```

为了用密钥以后可以使用生成的密钥来加密、解密以及做签名等运算，我们要把生成的密钥保存起来。

```
pri_key=RSAPrivateKey_dup(rsa);//提取私钥信息
out=BIO_new_file("rsa_pri.key","w");
//保存私钥信息
ret=PEM_write_bio_RSAPrivateKey(out,pri_key,enc,NULL,0,NULL,"123456");
pub_key=RSAPublicKey_dup(rsa);  //提取公钥信息
out=BIO_new_file("rsa_pub.key","w");
ret=PEM_write_bio_RSAPublicKey(out,pub_key);//保存公钥信息
```

2. 对某个文件 Hash 后计算其数字签名，并把得到的签名信息保存到文件。

1) 读取私钥信息，用以做签名。

```
 OpenSSL_add_all_algorithms();
in=BIO_new_file("rsa_pri.key","rb");
pri_key=PEM_read_bio_RSAPrivateKey(in,&pri_key,NULL,"123456");
```

2) 读取待签名的文件。

```
  BIO   *in;
int fd, len;
fd = open("hello.txt", O_RDWR);
while((i=read(fd,data,DATA_MAX_LEN))>0)
{
        filelen=i;
}
close(fd);
data[filelen]='\0';
```

3) 计算文件的 Hash 值，并利用私钥进行签名。

```
MD5(data,strlen((const char *)data),hashdata);
if(1!=(RSA_sign(nid,hashdata,55,signdata,&signlen,pri_key)))
{//对 Hash 后的值进行签名
        printf("RSA_sign err\n");
}
signdata[signlen]='\0';
//下面把签名的值写入文件
fd = open("hello.sign", O_CREAT | O_RDWR, S_IRUSR | S_IWUSR);
if(fd)
{
        write(fd,signdata);
}
else
{
        printf("open hello.sign err!\n");
}
close(fd);
```

3．利用公钥对所生成的数字签名进行验证。

1) 读取文件并做 Hash 值

```
fd = open("hello.txt", O_RDWR);
while((i=read(fd,data,DATA_MAX_LEN))>0)
{
        filelen=i;
}
close(fd);
data[MD5(data,strlen((const char *)data),hashdata);filelen]='\0';
```

2) 读取公钥并做验证

```
 in=BIO_new_file("rsa_pub.key","r");
pub_key=PEM_read_bio_RSAPublicKey(in, NULL, NULL, NULL);
if(pub_key==NULL)
{
        printf("read public key err!\n");
        return -1;
}
fd = open("hello.sign", O_RDWR);
while((i=read(fd,signdata,DATA_MAX_LEN))>0)
{
        filelen=i;
}
signdata[filelen]='\0';
if(1!=RSA_verify(nid,hashdata,55,signdata,filelen/3,pub_key))
{
        printf("RSA_verify err!\n");
        return -1;
}
printf("\nRSA_verify OK!\n");
```