

# 实验1：私有CA证书签发的简单实现

学号：2112066

姓名：于成俊

专业：密码科学与技术

## 一、前期准备

(1) 在VMware Workstation Pro上安装Ubuntu 20.04.6 LTS系统

(2) 安装成功后，在Ubuntu的命令行输入以下命令安装OpenSSL工具

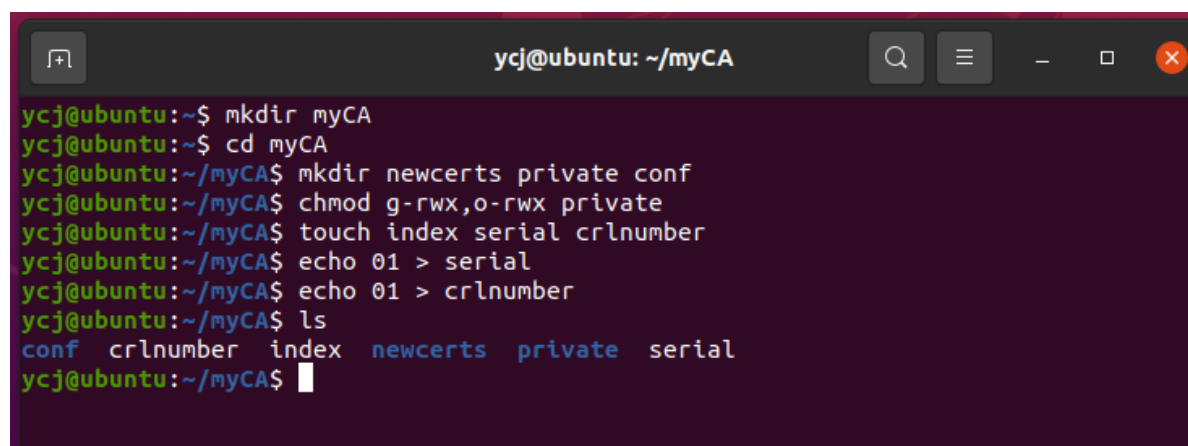
- `sudo apt-get update`：更新本地软件包列表
- `sudo apt-get install openssl`：安装 OpenSSL 软件包

## 二、实验内容

### (1) 搭建私有CA

1.使用以下指令创建私有CA所需要的文件目录，保存CA的相关信息

- `mkdir myCA`：创建CA根文件夹
- `cd myCA`：进入CA根文件夹
- `mkdir newcerts private conf`：创建三个文件夹，用来存放新发放证书、私钥和配置文件
- `chmod g-rwx,o-rwx private`：设置private文件夹的操作权限
- `touch index serial crlnumber`：创建证书信息数据库、证书序号文件、crl序号文件
- `echo 01 > serial`：初始化证书的序号
- `echo 01 > crlnumber`：初始化吊销证书序号

A terminal window titled 'ycj@ubuntu: ~/myCA' showing the execution of commands to create a private CA directory structure. The commands and their outputs are as follows:

```
ycj@ubuntu:~$ mkdir myCA
ycj@ubuntu:~$ cd myCA
ycj@ubuntu:~/myCA$ mkdir newcerts private conf
ycj@ubuntu:~/myCA$ chmod g-rwx,o-rwx private
ycj@ubuntu:~/myCA$ touch index serial crlnumber
ycj@ubuntu:~/myCA$ echo 01 > serial
ycj@ubuntu:~/myCA$ echo 01 > crlnumber
ycj@ubuntu:~/myCA$ ls
conf crlnumber index newcerts private serial
ycj@ubuntu:~/myCA$
```

使用以下命令可以查看证书序号文件和crl序号文件

- vim serial

[illegible]

- vim crlnumber

[illegible]

## 2.创建生成CA自签名证书的配置文件

- 使用 `cd conf` 命令进入配置文件夹
- 使用 `vim genca.conf` 创建用来生成自签名证书的配置文件

```
ycj@ubuntu:~/myCA$ cd conf
ycj@ubuntu:~/myCA/conf$ vim genca.conf
```

- 将配置文件设置如下:

```
[ req ]
default_keyfile = /home/ycj/myCA/private/cakey.pem
default_md = md5
prompt = no
distinguished_name = ca_distinguished_name
x509_extensions = ca_extensions

[ ca_distinguished_name ]
organizationName = DCYorg
organizationalUnitName = DCYunit
commonName = DCY
emailAddress = dcy@nankai.edu.cn

[ ca_extensions ]
basicConstraints = CA:true

~
~
~
~
~
~
~
"genca.conf" 16L, 354C
```

- 解析如下

```
[ req ]
default_keyfile = /home/ycj/myCA/private/cakey.pem
#注意修改为存放CA私钥的实际路径
default_md = md5
# 指定创建证书请求时对申请者信息进行数字签名的单向加密算法
prompt = no
# 值为no时, 不提示输入证书请求的字段信息, 直接从openssl.cnf中读取 distinguished_name =
ca_distinguished_name
# 扩展属性段落, 用于指定证书请求时可被识别的字段名称
x509_extensions = ca_extensions
# 要添加到自签名证书的扩展字段
[ ca_distinguished_name ]
organizationName = DCYorg
organizationalUnitName = DCYunit
commonName = DCY
emailAddress = dcy@nankai.edu.cn

[ ca_extensions ]
basicConstraints = CA:true
```

# 用于决定证书是否可以作为CA证书使用，此处设置为true

### 3.生成私有CA的私钥和自签名证书（根证书）

- 使用 `openssl req -x509 -newkey rsa:2048 -out cacert.pem -outform PEM -days 2190 -config /home/ycj/myCA/conf/genca.conf` 命令生成x509自签名证书，这样，CA会按照 `gentestca.conf` 文件中配置的规则自签名生成证书。
- 结果如下：

```
ycj@ubuntu:~/myCA/conf$ openssl req -x509 -newkey rsa:2048 -out cacert.pem -outform PEM -days 2190 -config /home/ycj/myCA/conf/genca.conf
Generating a RSA private key
.....+++++
.....+++++
writing new private key to '/home/ycj/myCA/private/cakey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
```

- 注意，这里有一个需要注意的地方，需要准确知道文件的绝对路径，否则会发生以下错误

```
ycj@ubuntu:~/myCA$ openssl req -x509 -newkey rsa:2048 -out cacert.pem -outform PEM -days 2190 -config /Home/ycj/myCA/conf/genca.conf
Can't open /Home/ycj/myCA/conf/genca.conf for reading, No such file or directory
140199013143872:error:02001002:system library:fopen:No such file or directory:../crypto/bio/bss_file.c:69:fopen('/Home/ycj/myCA/conf/genca.conf','r')
140199013143872:error:2006D080:BIIO routines: BIO_new_file:no such file:../crypto/bio/bss_file.c:76:
```

可使用 `readlink -f genca.conf` 命令来获取 `genca.conf` 文件的绝对路径，从而解决问题。

- 使用 `openssl x509 -in cacert.pem -text -noout` 命令查看CA自签名证书

```

ycj@ubuntu:~/myCA/conf$ openssl x509 -in cacert.pem -text -noout
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      37:a3:fe:81:5a:83:9a:4f:f9:96:5a:ca:30:24:88:8e:d2:74:84:ba
    Signature Algorithm: md5WithRSAEncryption
    Issuer: O = DCYorg, OU = DCYunit, CN = DCY, emailAddress = dcy@nankai.edu.cn
    Validity
      Not Before: Feb 27 07:11:46 2024 GMT
      Not After : Feb 25 07:11:46 2030 GMT
    Subject: O = DCYorg, OU = DCYunit, CN = DCY, emailAddress = dcy@nankai.edu.cn
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public-Key: (2048 bit)
      Modulus:
        00:de:d5:f0:4a:17:a3:cc:03:e6:6c:03:46:3b:39:
        8d:65:56:11:ba:e7:80:e2:d5:a9:49:24:67:7b:9d:
        92:17:87:b8:5c:0d:a1:0d:94:c4:92:44:ef:3a:e5:
        b4:0a:5f:a2:1c:d5:e3:fc:c7:30:be:ef:08:5f:a4:
        c0:b5:17:8d:22:94:0c:15:37:ab:a8:51:85:8d:ef:
        4b:1e:7f:d8:23:ef:18:0e:e8:21:ff:40:e4:c2:bd:
        97:7b:db:00:88:ef:f1:28:a6:3f:32:49:c5:f7:3e:
        43:f1:f8:61:bd:0c:34:51:e8:7a:3f:cd:d1:ae:1b:
        25:e4:37:ba:24:25:19:46:63:91:0e:00:7c:53:44:
        d2:25:ab:ce:5f:05:68:1e:5e:67:23:8d:d9:a1:63:
        c8:7d:07:8f:b3:ae:a2:53:a0:6b:4b:a7:77:e3:e6:
        c7:0a:cd:f2:b1:54:7b:c6:05:50:8a:e9:9d:83:76:
        42:33:67:38:e0:59:30:7c:0c:54:57:c7:18:a1:3e:
        26:0e:7c:ec:13:47:05:3f:95:55:36:d2:e3:d2:08:
        1f:e0:4e:e4:a3:e6:80:0e:97:95:fb:44:df:65:32:
        42:19:e9:7d:8b:c5:35:60:b8:e5:a8:44:c5:a6:2f:
        87:91:70:00:a2:30:cb:40:a3:5d:23:99:9a:54:30:
        d6:9d
      Exponent: 65537 (0x10001)
    X509v3 extensions:
      X509v3 Basic Constraints:
        CA:TRUE
    Signature Algorithm: md5WithRSAEncryption
    50:96:c6:55:6f:c9:22:68:b5:71:26:e1:79:20:4d:90:85:24:
    36:97:2b:52:68:45:64:61:b9:70:0c:12:c2:63:4e:61:81:ab:
    99:cc:0c:a3:94:d5:7c:38:44:db:a2:68:71:27:19:3c:bb:2e:
    2b:10:83:e4:ee:43:c4:b3:71:89:26:c2:96:f1:ce:d5:47:0c:
    68:1c:5e:a7:ed:e3:b6:dc:08:be:64:9e:95:0d:a3:bb:38:72:
    e9:bf:ff:c1:98:7e:d3:69:9c:f8:64:10:c8:a5:4f:3c:66:00:
    1d:ea:2e:4e:de:59:47:bd:7a:49:8c:b6:1a:68:9a:6f:4a:6b:
    a8:a6:37:be:9f:12:dd:76:bb:96:8e:b3:e1:35:3b:1b:cd:c0:
    14:be:d8:e1:df:dd:99:98:bb:69:96:0e:74:d5:6f:69:b5:a2:
    1d:37:01:cc:f1:75:f2:f7:51:ab:43:32:6b:bd:a5:bb:f8:70:
    e2:74:94:06:f0:15:1f:33:53:8f:83:ef:28:1d:a0:b0:8d:c0:
    ca:54:a7:d2:ea:80:bc:2f:19:79:37:98:76:4f:49:6b:d1:a7:
    aa:5a:47:03:77:73:0a:ab:94:09:66:03:f6:7b:c9:03:c2:45:
    06:59:43:c9:b6:d3:9e:a4:21:82:07:36:e2:d3:b4:16:1d:8d:
    72:96:34:77

```

## (2) 私有CA为服务器签发证书

### 1.创建用来为其他请求签发证书的配置文件

- 使用 `cd conf` 命令再次进入配置文件夹
- 使用 `vim ca.conf` 命令创建用来为其他请求签发证书的配置文件
- 配置文件设置如下:

```
[ ca ]
default_ca = myCA

[ myCA ]
dir = /home/ycj/myCA
crl_dir = $dir/conf
database = $dir/index
new_certs_dir = $dir/newcerts

certificate = $dir/conf/cacert.pem
serial = $dir/serial
private_key = $dir/private/cakey.pem
crlnumber = $dir/crlnumber
RANDFILE = $dir/private/.rand

default_days = 365
default_crl_days = 30
default_md = md5
unique_subject = no

policy = policy_any

[ policy_any ]

stateOrProvinceName = optional
localityName = optional
organizationName = optional
organizationalUnitName = optional
commonName = supplied
emailAddress = optional
```

- 解析如下：

[ ca ]：指定了CA的配置段。

default\_ca = myCA：设置默认的CA名称为 myCA。

[ myCA ]：定义了名为 myCA 的CA的配置段。

dir = /home/ycj/myCA：指定CA的工作目录为 /home/ycj/myCA

crl\_dir = \$dir/conf：指定证书撤销列表（CRL）的存储目录为 \$dir/conf。

database = \$dir/index：指定了CA数据库的位置。

new\_certs\_dir = \$dir/newcerts：指定新证书存储的目录。

certificate = \$dir/conf/cacert.pem：指定CA证书文件的位置。

serial = \$dir/serial：指定证书序列号文件的位置。

private\_key = \$dir/private/cakey.pem：指定CA私钥文件的位置。

crlnumber = \$dir/crlnumber：指定CRL编号文件的位置。

RANDFILE = \$dir/private/.rand：指定随机数文件的位置。

default\_days = 365: 设置默认证书有效期为365天。

default\_crl\_days = 30: 设置默认CRL有效期为30天。

default\_md = md5: 设置默认的摘要算法为MD5（不推荐，因为MD5已经不安全）。

unique\_subject = no: 允许重复的主题名称。

[ policy\_any ]: 定义了一个名为 policy\_any 的策略。

下面的字段指定了对证书中各个信息字段的要求，比如州/省、城市、组织等，以及它们是否是可选的或必需的。

```
stateOrProvinceName = optional
localityName = optional
organizationName = optional
organizationalUnitName = optional
commonName = supplied
emailAddress = optional
```

## 2. 模拟服务器，生成私钥与证书申请请求文件

- 使用 `mkdir server` 命令在任意路径下创建服务器文件夹server
- 使用 `openssl req -new -newkey rsa:1024 -keyout server.key -out serverreq.pem -subj "/O=ServerCom/OU=ServerOU/CN=server"` 命令生成server的1024位私钥server.key和证书申请的请求文件serverreq.pem

结果如下：

```
ycj@ubuntu:~$ openssl req -new -newkey rsa:1024 -keyout server.key -out serverreq.pem -subj "/O=ServerCom/OU=ServerOU/CN=server"
Generating a RSA private key
.....+++++
.....+++++
writing new private key to 'server.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
```

在这里，我输入实验指导书上的命令 `openssl req -newkey rsa:1024 -keyout server.key -out serverreq.pem -subj "/O=ServerCom/OU=ServerOU/CN=server"`，发现不识别，上网搜索，发现是版本可能不同，所以加了个-new

## 3. CA根据服务器的证书请求文件生成证书并将其返回给服务器

- 使用命令 `openssl ca -in serverreq.pem -out server.crt -config /home/ycj/myCA/conf/ca.conf` 向私有CA提交证书请求文件serverreq.pem，CA生成并返回证书server.crt，注意，这里生成证书的规则是参照之前为CA定义的ca.conf配置文件执行的
- 结果如下：

```
ycj@ubuntu:~$ openssl ca -in serverreq.pem -out server.crt -config /home/ycj/myCA/conf/ca.conf
Using configuration from /home/ycj/myCA/conf/ca.conf
Enter pass phrase for /home/ycj/myCA/private/cakey.pem:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
organizationName      :ASN.1 12:'ServerCom'
organizationalUnitName:ASN.1 12:'ServerOU'
commonName            :ASN.1 12:'server'
Certificate is to be certified until Feb 26 08:01:55 2025 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
```



- 可以在index文件中看到证书信息数据库更新

V	250226080155Z	01	unknown /O=ServerCom/OU=ServerOU/CN=server
---	---------------	----	--

- 证书目录中也生成了新证书备份，如下：

```
ycj@ubuntu:~/myCA/newcerts$ ls
01.pem
```

### (3) 私有CA为客户端签发证书

私有CA为客户端签发证书与（2）私有CA为服务器签发证书的步骤一样，在（2）配置文件的基础上，执行以下步骤

#### 1.模拟客户端，生成私钥与证书申请的请求文件

- 使用 `mkdir client` 命令在任意路径下创建服务器文件夹client
- 使用 `openssl req -new -newkey rsa:1024 -keyout client.key -out clientreq.pem -subj "/O=ClientCom/OU=ClientOU/CN=client"` 命令生成client的1024位私钥client.key和证书申请的请求文件clientreq.pem

结果如下：

```
ycj@ubuntu:~$ openssl req -new -newkey rsa:1024 -keyout client.key -out clientreq.pem -subj "/O=ClientCom/OU=ClientOU/CN=client"
Generating a RSA private key
.....+++++
writing new private key to 'client.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
```

#### 2. CA根据服务器的证书请求文件生成证书并将其返回给服务器

- 使用命令 `openssl ca -in clientreq.pem -out client.crt -config /home/ycj/myCA/conf/ca.conf` 向私有CA提交证书请求文件clientreq.pem，CA生成并返回证书client.crt，注意，这里生成证书的规则是参照之前为CA定义的ca.conf配置文件执行的
- 结果如下：

```
ycj@ubuntu:~$ openssl ca -in clientreq.pem -out client.crt -config /home/ycj/myCA/conf/ca.conf
Using configuration from /home/ycj/myCA/conf/ca.conf
Enter pass phrase for /home/ycj/myCA/private/cakey.pem:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
organizationName      :ASN.1 12:'ClientCom'
organizationalUnitName:ASN.1 12:'ClientOU'
commonName            :ASN.1 12:'client'
Certificate is to be certified until Feb 28 00:03:19 2025 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
```

- 可以在index文件中看到证书信息数据库更新

V	250226080155Z	01	unknown /O=ServerCom/OU=ServerOU/CN=server
V	250228000319Z	02	unknown /O=ClientCom/OU=ClientOU/CN=client

- 证书目录中也生成了新证书备份，如下：

```
ycj@ubuntu:~/myCA/newcerts$ ls
01.pem 02.pem
```



## (4) CA吊销用户证书

### 1.生成证书吊销列表 (CRL)

- 在之前生成的ca.conf配置文件基础上, 使用命令 `openssl ca -config /home/ycj/myCA/conf/ca.conf -gencrl -out ca.crl -crl days 30` 生成一个证书吊销列表, 并将其输出到 `ca.crl` 文件中
- 结果如下:

```
ycj@ubuntu:~$ openssl ca -config /home/ycj/myCA/conf/ca.conf -gencrl -out ca.crl -crl days 30
Using configuration from /home/ycj/myCA/conf/ca.conf
Enter pass phrase for /home/ycj/myCA/private/cakey.pem:
```

### 2.使用openssl ca指令完成用户证书吊销, 将对应证书的序列号添加到CRL中

- 在index文件中, 找到想吊销证书的对应编号

V	250226080155Z	01	unknown /O=ServerCom/OU=ServerOU/CN=server
V	250228000319Z	02	unknown /O=ClientCom/OU=ClientOU/CN=client

- 在这里, 我们吊销01证书, 使用命令 `openssl ca -revoke /home/ycj/myCA/newcerts/01.pem -config "/home/ycj/myCA/conf/ca.conf"` 进行吊销
- 结果如下:

```
ycj@ubuntu:~$ openssl ca -revoke /home/ycj/myCA/newcerts/01.pem -config "/home/ycj/myCA/conf/ca.conf"
Using configuration from /home/ycj/myCA/conf/ca.conf
Enter pass phrase for /home/ycj/myCA/private/cakey.pem:
Revoking Certificate 01.
Data Base Updated
```

### 3.更新并重新生成CRL

- 使用命令 `openssl ca -gencrl -out /home/ycj/myCA/ca.crl -config /home/ycj/myCA/conf/ca.conf -crl days 30` 更新证书吊销列表

如下:

```
ycj@ubuntu:~$ openssl ca -gencrl -out /home/ycj/myCA/ca.crl -config /home/ycj/myCA/conf/ca.conf -crl days 30
Using configuration from /home/ycj/myCA/conf/ca.conf
Enter pass phrase for /home/ycj/myCA/private/cakey.pem:
```

- 还可以在crlNumber文件中看到crl序号更新:



- 使用命令 `openssl crl -in ca.crl -noout -text` 可以查看 `crl` 文件

```
ycj@ubuntu:~/myCA$ openssl crl -in ca.crl -noout -text
Certificate Revocation List (CRL):
    Version 2 (0x1)
    Signature Algorithm: md5WithRSAEncryption
    Issuer: O = DCYorg, OU = DCYunit, CN = DCY, emailAddress = dcy@nankai.edu.cn
    Last Update: Feb 29 02:08:49 2024 GMT
    Next Update: Mar 30 02:08:49 2024 GMT
    CRL extensions:
        X509v3 CRL Number:
            3
Revoked Certificates:
    Serial Number: 01
    Revocation Date: Feb 29 01:54:51 2024 GMT
    Signature Algorithm: md5WithRSAEncryption
    0a:7c:3e:43:55:10:3b:d0:2d:12:67:e7:89:4e:0c:3f:e4:24:
    19:76:30:b8:8a:48:64:f5:07:b8:55:20:ce:87:8e:20:7e:c2:
    8f:87:d7:54:ab:08:df:d7:82:3e:13:5e:bf:e7:ce:8d:6a:a1:
    f2:ea:5b:8a:ba:f4:c7:c8:a0:9e:87:de:2d:26:1f:3a:e3:b7:
    38:f8:46:5e:72:f9:e8:60:21:08:46:1e:31:d0:7d:b8:aa:9e:
    a0:e2:5a:fa:e2:0c:7f:79:ff:38:a7:11:f0:a2:32:be:72:95:
    17:3d:c4:41:00:48:30:e3:73:1f:51:1d:55:05:94:8a:51:9f:
    64:f8:5d:13:71:79:a0:11:14:f0:5b:f1:78:6c:16:ba:2b:42:
    57:45:05:73:2e:45:ef:3d:6d:87:c9:c4:8e:5b:6f:f0:db:d2:
    f4:ae:e8:63:df:13:15:e3:1a:5b:89:ee:4a:5a:63:27:e9:22:
    46:c5:6c:19:d5:73:85:f7:b0:1f:af:43:99:f8:cf:e2:11:c0:
    02:97:7a:ca:a7:b5:76:ff:24:f6:89:19:98:2d:02:50:a5:a5:
    be:41:5c:2a:9c:06:6d:6c:a4:bc:87:1b:9a:37:5f:a9:3b:e6:
    7a:1f:a2:71:c2:69:a7:f1:85:10:9c:c9:19:b1:8a:f6:a0:02:
    ac:7f:c4:25
```

### 三、思考题

#### CA如何验证证书的有效性？需要考虑到哪些方面？

CA（证书颁发机构）验证证书的有效性是通过一系列步骤来完成的，主要包括以下几点：

1. **验证签名的有效性**：CA 首先检查证书中的签名是否有效。这是通过使用 CA 的公钥来验证签名的方式完成的。如果签名有效，则表明该证书确实是由 CA 签发的。
2. **检查证书是否已过期**：CA 验证证书中的有效期，确保当前时间位于证书的有效期内。如果证书已经过期，则被认为无效。
3. **检查证书吊销状态**：CA 验证证书的吊销状态，以确定该证书是否已被吊销。证书吊销列表（CRL）是一个包含被吊销证书序列号的列表。CA 将验证证书序列号是否出现在 CRL 中，以确定证书的状态。
4. **验证证书链**：对于服务器证书，客户端通常需要验证服务器证书的完整性。这包括验证服务器证书是否由受信任的 CA 签发，并且是否能够追溯到根证书。这通常通过检查证书链中的所有证书来完成，确保每个证书都被一个信任的 CA 签发，直到达到根证书。
5. **检查证书是否被撤销**：CA 会定期发布 CRL，其中包含了已经被吊销的证书的列表。在验证证书时，客户端可能会检查 CRL 来确定证书是否已经被吊销。

通过这些步骤，CA 可以验证证书的有效性，并确保其可信度和安全性。这种验证过程对于建立安全的通信和保护网络资源至关重要。

### 四、实验总结

在本次实验中，我学习了一些关于证书颁发机构（CA）和证书管理的基本指令。以下是在实验中针对指令的主要心得体会：

1. **生成自签名证书**：使用 OpenSSL 命令生成自签名证书是学习证书管理的重要一步。通过生成自签名证书，我了解了证书的基本结构和生成过程。
2. **签发证书请求**：使用 OpenSSL 的 `openssl req` 命令可以生成证书请求文件，这是向 CA 请求证书的第一步。了解如何生成证书请求对于获得有效证书至关重要。

3. **签发证书：**通过 OpenSSL 的 `openssl ca` 命令，CA 可以签发证书并将其颁发给请求者。了解如何使用 `openssl ca` 命令是学习证书管理的关键一步。
4. **生成证书吊销列表（CRL）：**使用 OpenSSL 的 `openssl ca -gencrl` 命令可以生成证书吊销列表，用于列出已被吊销的证书序列号。了解如何生成和管理 CRL 对于保证证书系统的安全至关重要。
5. **更新证书吊销列表（CRL）：**定期更新 CRL 是保证证书系统安全的重要步骤之一。通过学习如何更新 CRL，我认识到了维护证书系统的重要性。

总的来说，通过实践这些指令，我更加深入地理解了证书管理的流程和原理。掌握这些基本指令对于构建安全的证书系统和保障通信安全具有重要意义。