制作第一张自己的SSL证书

准备工具

安装keytool（jdk自带）、openssl、nginx、web服务

有两种方法生成数字证书，一种是用JDK带的keytool，另一种是用openssl。

我们将利用openssl完成以下表格内容：

完成项	输出文件
CA服务器根证书	cacert.pem 证书
服务器证书	servercert.pem 证书 serverkey.pem 私钥
客户端证书	clientcert.pem 证书 clientkey.pem 私钥

先做一个服务端单向认证，最后完成一个服务端客户端双向认证。

制作根证书

制作一个CA私服，取名叫testca。

准备CA工作目录

```
mkdir "$HOME/testca"
cd "$HOME/testca"
mkdir newcerts private conf
chmod g-rwx,o-rwx private
echo "01" > serial
touch index.txt
```

$HOME/testca为待建CA的主目录

newcerts子目录将存放CA签署（颁发）过的数字证书（证书备份目录）

private目录用于存放CA的私钥

conf只是用于存放一些简化参数用的配置文件

serial和index.txt分别用于存放下一个证书的序列号和证书信息数据库

生成根证书

配置根证书

创建testca根证书配置文件及内容

```
vi "$HOME/testca/conf/gentestca.conf"
```

内容如下

```
####################################
[ req ]
default_keyfile = $ENV::HOME/testca/private/cakey.pem
default_md = md5
prompt = no
distinguished_name = ca_distinguished_name
x509_extensions = ca_extensions

[ ca_distinguished_name ]
organizationName = dongnaoedu
organizationalUnitName  = dongnao
commonName = ca.dongnaoedu.com
emailAddress = 805921455@qq.com

[ ca_extensions ]
basicConstraints = CA:true
####################################
```

申请根证书

CA服务器，自己给自己颁发证书，生成CA的私钥和自签名证书，即根证书。

```
cd "$HOME/testca"
openssl req -x509 -newkey rsa:2048 -out cacert.pem -outform PEM -days 2190 -
config "$HOME/testca/conf/gentestca.conf"
# req 表示发起一个证书签名请求
# -x509 用x509结构替代cert
# -nwekey rsa:2048，新建一个2048 bit的rsa秘钥
# -out 输出的证书文件
# -outform 输出的格式，DER或者PEM
# -days 有效时间2190天
# -config 请求的配置文件，这里是根证书的配置信息
```

执行过程中需要输入CA私钥的保护密码，假设我们输入密码： 123456

查看一下自己CA证书的内容

```
openssl x509 -in cacert.pem -text -noout
```

准备根证书配置文件

方便以后用它来生成，在它下面的子证书

```
vi "$HOME/testca/conf/testca.conf"
```

写入文件内容

```
####################################
[ ca ]
default_ca      = testca                # The default ca section

[ testca ]
dir             = $ENV::HOME/testca         # top dir
```

```
database        = $dir/index.txt          # index file.
new_certs_dir   = $dir/newcerts           # new certs dir

certificate     = $dir/cacert.pem         # The CA cert
serial          = $dir/serial             # serial no file
private_key     = $dir/private/cakey.pem  # CA private key
RANDFILE        = $dir/private/.rand      # random number file

default_days    = 365                     # how long to certify for
default_crl_days= 30                      # how long before next CRL
default_md      = md5                     # message digest method to use
unique_subject = no                       # Set to 'no' to allow creation of
                                          # several ctificates with same subject.
policy          = policy_any              # default policy

[ policy_any ]
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional

########################################
```

单向认证

我们可以用openssl为服务器或用户生成公钥密钥，并用上面创建的CA根证书cacert.pem签发对应的私钥（密钥）的数字证书。

准备目录

我们把服务器相关的东西生成到CA的$HOME/testca/test/server目录里。

```
mkdir -p "$HOME/testca/test/server"
cd "$HOME/testca/test/server"
```

创建服务器私钥，并生成testca的证书请求文件

通过req命令，生成一个签名证书申请，使用rsa:1024生成新秘钥，申请证书为testkey.pem，格式为PEM，证书签名参数为-subj指定的内容

```
openssl req -newkey rsa:1024 -out serverreq.pem -keyout serverkey.pem -keyform
PEM -outform PEM  -subj "/O=ABCom/OU=servers/CN=servername"

# req 表示发起一个证书签名请求
# -nwekey rsa:1024，新建一个1024 bit的rsa秘钥
# -out 输出的证书请求文件
# -keyout 表示存放生成私钥的文件
# -keyform 表示生成的秘钥文件格式，这里是PEM
# -outform 输出的格式，DER或者PEM
# -subj 设置或修改请求证书签名主题
```

执行命令过程中输入密钥保护密码，我们输入：949494

serverkey.pem为的私钥，serverreq.pem为CA签名证书请求文件。

查看请求文件内容

```
openssl req -in serverreq.pem -text -noout
```

CA签发证书

```
openssl ca -in serverreq.pem -out servercert.pem -config
"$HOME/testca/conf/testca.conf"
```

执行过程中需要输入CA私钥的保护密码，前面设置的123456。

查看证书内容

```
openssl x509 -in servercert.pem -text -noout
```

Nginx配置

找到nginx运行的配置文件，将证书servercert.pem及私钥serverkey.pem放到配置目录下，修改内容如下

```
http {
    # 配置https服务
    server {
        # 开启443端口，ssl安全协议
        listen       443 ssl;
        server_name  localhost;

        # 配置证书及服务器私钥
        ssl_certificate      servercert.pem;
        ssl_certificate_key  serverkey.pem;

        # 会话参数的缓存,所有工作进程之间共享的缓存
        ssl_session_cache    shared:SSL:1m;
        ssl_session_timeout  5m;
        ssl_protocols TLSv1 TLSv1.1 TLSv1.2;

        # 启用的密码
        ssl_ciphers  ECDHE-RSA-AES128-GCM-SHA256:HIGH:!aNULL:!MD5:!RC4:!DHE;
        # SSLv3和TLS协议时,服务器密码优先于客户端密码
        ssl_prefer_server_ciphers  on;

        location / {
            root   /data/www/;
            index  welcome.html;
        }
    }
}
```

双向认证

前面我们已经制作了服务器的证书，下面我只需要制作客户端证书就可以了。

创建客户端证书

用openssl创建客户端证书，只有这些CA签发的客户端证书才被服务器信任，才能通过HTTPS访问服务器。这就是“HTTPS服务器验证客户端证书”的关键配置。讲

```
mkdir -p "$HOME/testca/test/client"
cd "$HOME/testca/test/client"

openssl req -newkey rsa:1024 -keyout clientkey.pem -keyform PEM -out
clientreq.pem -outform PEM  -subj "/O=client/OU=client/CN=client"

openssl ca -in clientreq.pem -out clientcert.pem -config
"$HOME/testca/conf/testca.conf"
```

设定执行过程中的密码都是：654321

制作PKCS12格式的客户端证书

PKCS12格式的证书包含私钥和公钥内容，它是描述个人信息交换的语法标准。描述了将用户公钥、私钥、证书和其他相关信息打包的语法或格式。我们制作的这个PKCS#12文件将包含密钥、证书和颁发该证书的CA证书。

```
openssl pkcs12 -export -in clientcert.pem -inkey clientkey.pem -out client.p12 -
name hash -CAfile "$HOME/testca/cacert.pem"
```

我们将导出密码也设置为：654321

将生成的.p12文件导入到浏览器个人证书列表或USB秘钥U盘中，客户端就能用它来通过服务的验证。

Nginx配置

参考资料

找到nginx运行的配置文件，将服务器证书servercert.pem、钥serverkey.pem、ca根证书cacert.pem放到配置目录下，修改内容如下

```
http {
    # 配置https服务
    server {
        # 开启443端口，ssl安全协议
        listen       443 ssl;
        server_name  localhost;

        # 配置证书及服务器私钥
        ssl_certificate      servercert.pem;
        ssl_certificate_key  serverkey.pem;
```

```
        ssl_client_certificate  cacert.pem; # 根级证书公钥，用于验证各个二级client
        ssl_verify_client on;   #开启客户端证书验证

        # 会话参数的缓存,所有工作进程之间共享的缓存
        ssl_session_cache    shared:SSL:1m;
        ssl_session_timeout  5m;
        ssl_protocols TLSv1 TLSv1.1 TLSv1.2;

        # 启用的密码
        ssl_ciphers  ECDHE-RSA-AES128-GCM-SHA256:HIGH:!aNULL:!MD5:!RC4:!DHE;
        # SSLv3和TLS协议时,服务器密码优先于客户端密码
        ssl_prefer_server_ciphers  on;

        location / {
            root   /data/www/;
            index  welcome.html;
        }
    }
}
```

CA的日常操作

签发证书

假设收到一个证书请求文件名为req.pem，文件格式应该是PKCS#10格式（标准证书请求格式）

先查看证书请求的内容

```
openssl req -in req.pem -text -noout
```

签发证书

```
openssl ca -in req.pem -out cert.pem -config "$HOME/testca/conf/testca.conf"
```

执行过程中会要求输入访问CA的私钥密码，前面设置的123456

命令执行完毕，cert.pem就是签发好的证书，在$HOME/testca/newcerts里也会有一个相同的证书副本，文件名为证书序列号。

查看生成的证书的内容

```
openssl x509 -in cert.pem -text -noout
```

作废证书

由于用户私钥泄露或其他情况，需要吊销一个未过期的证书。假设需要被吊销的证书文件为cert.pem，则执行以下命令吊销证书：

```
openssl ca -revoke cert.pem -config "$HOME/testca/conf/testca.conf"
```

生成证书作废列表CRL

公开被吊销的证书列表，可以生成证书吊销列表（CRL），执行命令如下

```
openssl ca -gencrl -out testca.crl -config "$HOME/testca/conf/testca.conf"
```

还可用-crldays和-crlhours参数来说明下一个吊销列表将在未来某个时候（多少天或多少小时后）发布。

查看检查testca.crl的内容

```
openssl crl -in testca.crl -text -noout
```

服务端如何检查该客户端证书是否已经被吊销？我们可以通过检查CRL（Certification Revocation List）即证书吊销列表来做这个工作。