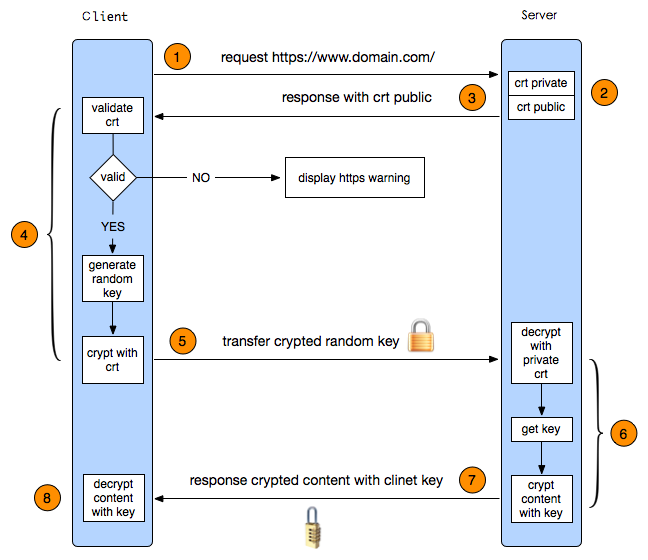
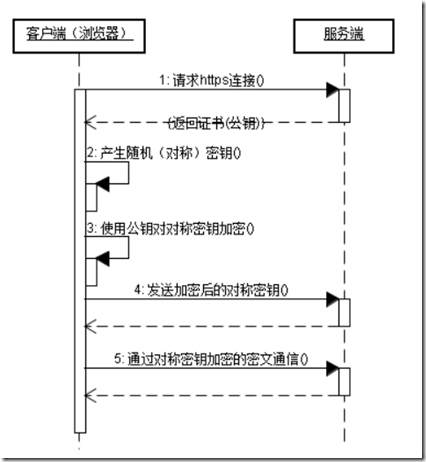
# ##HTTPS 要做什么###

* https 在传输层(TCP)和应用层之间增加了一层ssl ，从而将传输过程由明文变成密文
* 传输层通过端口来和应用层交换数据，因此同样的web应用只需要通过不同的端口来区分http 和 https，应用层的代码不需要任何变动

## \*加密\*

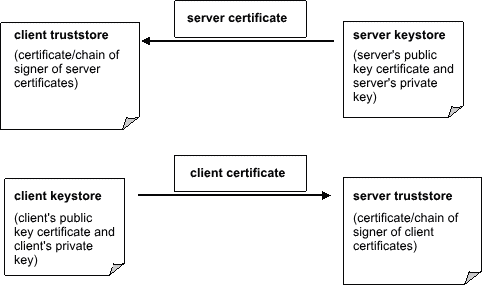




* 服务器生成公钥和私钥
* 把公钥放在证书里发送给客户端，私钥自己保存
* 客户端首先向一个权威的服务器检查证书的合法性，如果证书合法，客户端产生一段随机数，这个随机数就作为通信的密钥，我们称之为对称密钥，用公钥加密这段随机数，然后发送到服务器，如果客户端的truststore 库中已经加入了服务器的公钥就不需要权威的服务器检查证书的合法性。
* 服务器用密钥解密获取对称密钥，然后，双方就已对称密钥进行加密解密通信了
* 首次通信的时候才用非对称的方式加密解密

RSA性能是非常低的，原因在于寻找大素数、大数计算、数据分割需要耗费很多的CPU周期，所以一般的HTTPS连接只在第一次握手时使用非对称加密，通过握手交换对称加密密钥，在之后的通信走对称加密。

## \*双向认证\*



服务端用 keystore 存放受信任的客户端证书和私钥，客户端使用 truststore 存放受信任的服务器证书

- 客户端验证服务器发过来的证书是否在自己的信任证书库中，如果没有可以拒绝访问，也可以继续请求

- 服务器验证客户端传过来的证书（p12）是否在自己的信任证书库中，如果不在拒绝客户端的请求

# ###使用jdk的keytool生成相关证书###

测试服务器java 安装在

`cd /usr/java/jdk1.7.0\_45`

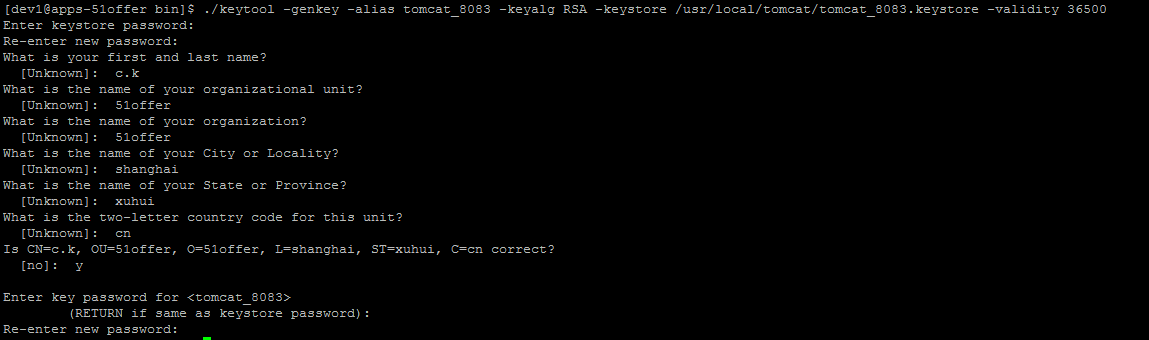
到bin目录下通过keytool 来生成相关证书

## ####生成服务器端证书库####

生成服务器端 keystore文件用于存放私钥，再用私钥加解密或者为别人做签名。

通过以下命令

`./keytool -genkey -alias tomcat\_8083 -keyalg RSA -keystore /usr/local/tomcat/tomcat\_8083.keystore -validity 36500`



`-validity 36500` 含义是证书有效期，36500表示100年，默认值是90天 `-alias tomcat\_8083`为自定义证书名称

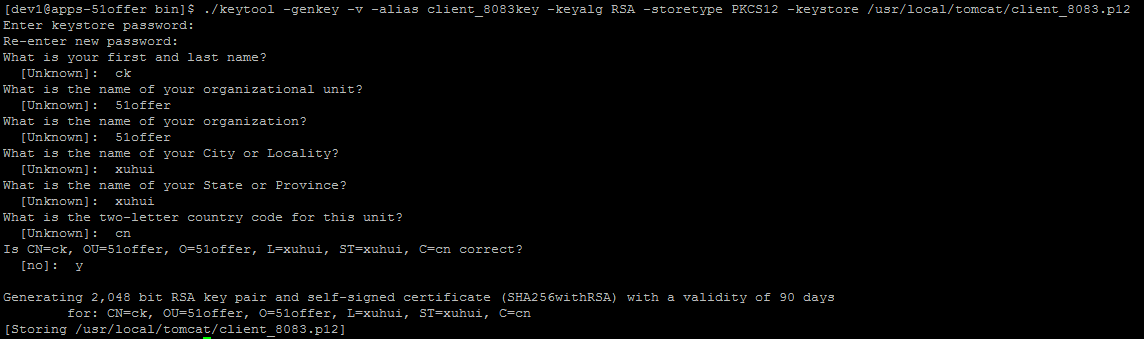
rsa 证书密码 111111

`/usr/local/tomcat/tomcat\_8083.keystore`

## ####生成客户端证书####

为浏览器生成证书，以便让服务器来验证它，这一步也可以认为是为客户端生成私钥。为了能将证书顺利导入至IE和Firefox，证书格式是PKCS12，因此，使用如下命令生成

`./keytool -genkey -v -alias client\_8083key -keyalg RSA -storetype PKCS12 -keystore /usr/local/tomcat/client\_8083.p12 `



## ####让服务器信任客户端证书####

由于是双向SSL认证，服务器必须要信任客户端证书，确保客户端的公钥证书是由服务器端颁发的，因此，必须把客户端证书添加为服务器的信任认证。由于不能直接将PKCS12格式的证书库导入，必须先把客户端证书导出为一个单独的CER文件，使用如下命令：

`./keytool -export -alias client\_8083key -keystore /usr/local/tomcat/client\_8083.p12 -storetype PKCS12 -storepass 111111 -rfc -file /usr/local/tomcat/client\_8083.cer `

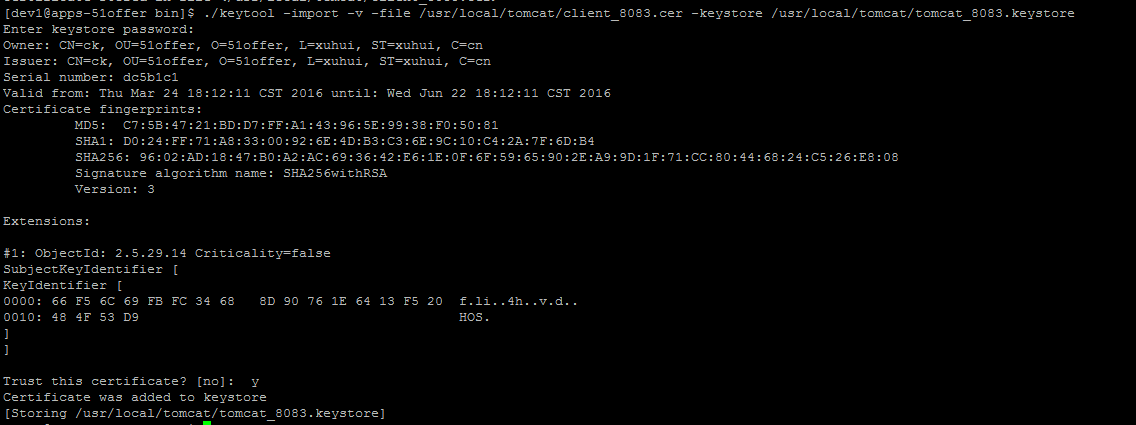
返回 `Certificate stored in file </usr/local/tomcat/client\_8083.cer> ` 表示成功。



(client\_8083key为自定义与客户端定义的client\_8083key要一致，111111是设置的密码)。通过以上命令，客户端证书就被我们导出到`client\_8083.cer`文件了。

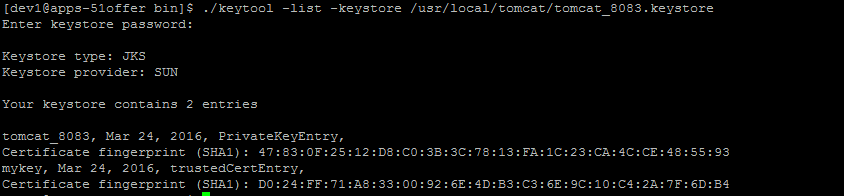
下一步，是将该文件导入到服务器的证书库，添加为一个信任证书使用命令如下：

`./keytool -import -v -file /usr/local/tomcat/client\_8083.cer -keystore /usr/local/tomcat/tomcat\_8083.keystore `



通过list命令查看服务器的证书库，可以看到两个证书，一个是服务器证书(私钥)，一个是受信任的客户端证书，意思是只有客户端使用了` client.cer `证书才能够访问服务器。

`./keytool -list -keystore /usr/local/tomcat/tomcat\_8083.keystore`



## ####让客户端信任服务器证书####

既然是双向SSL认证，客户端也要验证服务器证书。如果客户端有了服务端的证书，那么浏览器就不会警告用户了，因此，必须把服务器证书添加到浏览的“受信任的根证书颁发机构”。由于不能直接将keystore格式的证书库导入，必须先把服务器证书导出为一个单独的CER文件，使用如下命令

`./keytool -keystore /usr/local/tomcat/tomcat\_8083.keystore -export -alias tomcat\_8083 -file /usr/local/tomcat/tomcat\_8083.cer `



通过以上命令，服务器证书就被我们导出到`/usr/local/tomcat/tomcat\_8083.cer`文件了。然后下载到本地，双击`tomcat\_8083.cer`文件，按照提示安装证书，将证书填入到\*受信任的根证书颁发机构\*。

## ####将服务器端证书导入到客户端的信任证书库####

上一步生成的 `tomcat\_8083.cer`用于浏览器，让浏览器认为服务端是受信任的站点。这一步生成的truststore文件保存的是一些可信任的证书，主要是java在代码中访问某个https的时候对被访问者进行认证的，以确保其实可信任的，下面使用HttpClient访问https网站的时候会用到。

truststore是必须的，如果我们没有显式的指定，那么java会默认指定为$JAVA\_HOME/lib/security/cacerts 这个文件。

如果要指定的话，可以在java的参数中进行指定：

```

-Djavax.net.ssl.keyStore=clientKeys

-Djavax.net.ssl.keyStorePassword=password

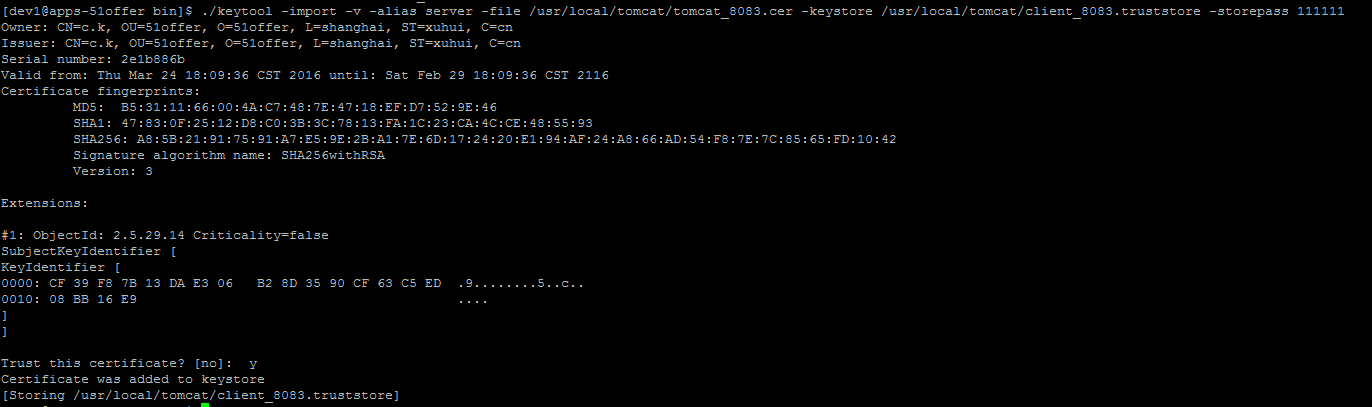
-Djavax.net.ssl.trustStore=clientTrust

-Djavax.net.ssl.trustStorePassword=password

```

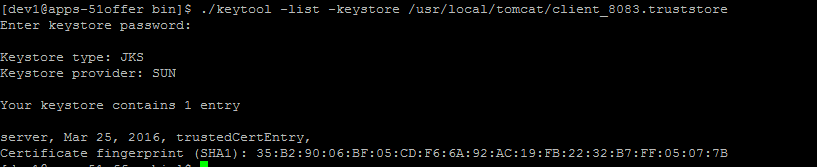
执行如下命令：

`./keytool -import -v -alias server -file /usr/local/tomcat/tomcat\_8083.cer -keystore /usr/local/tomcat/client\_8083.truststore -storepass 111111 `



执行下面命令查看 truststore 库中的证书

`./keytool -list -keystore /usr/local/tomcat/client\_8083.truststore `



## ####将服务器端证书导入到浏览器中####

双击 tomcat\_8083.cer 文件会有如下提示，点击\*是\*完成安装。



## ####配置Tomcat服务器####

打开Tomcat根目录下的`/conf/server.xml`，找到`Connector port="8443"`配置段，默认是被注释掉了，修改为如下：

```

<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"

SSLEnabled="true" maxThreads="150" scheme="https"

secure="true" clientAuth="true" sslProtocol="TLS"

keystoreFile="/usr/local/tomcat/tomcat\_8083.keystore" keystorePass="111111"

truststoreFile="/usr/local/tomcat/tomcat\_8083.keystore" truststorePass="111111" />

```

属性说明：

clientAuth:设置是否双向验证，默认为false，表示不需要验证客户端合法性，任何客户端都可以访问；设置为true，表示需要验证客户端发过来的证书（由服务器端生成的p12文件）是否在服务器端的 truststore 里面，如果没有就无法访问。

keystoreFile:服务器证书文件路径

keystorePass:服务器证书密码

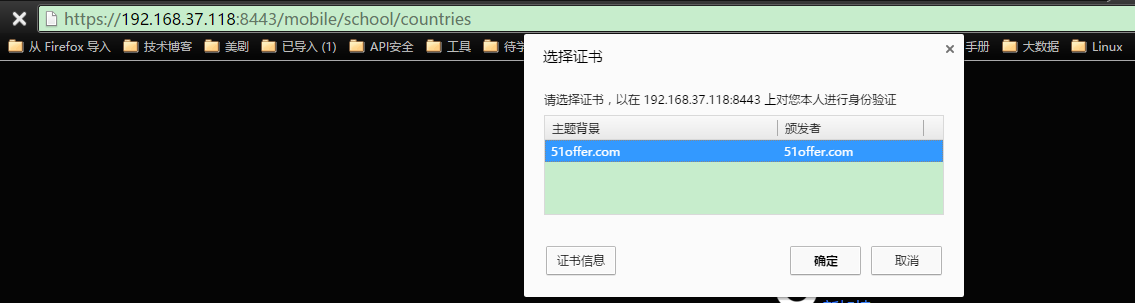
truststoreFile:用来验证客户端证书的根证书，此例中就是服务器证书

truststorePass:根证书密码

然后重启tomcat。

## ####使用浏览器访问https网站####

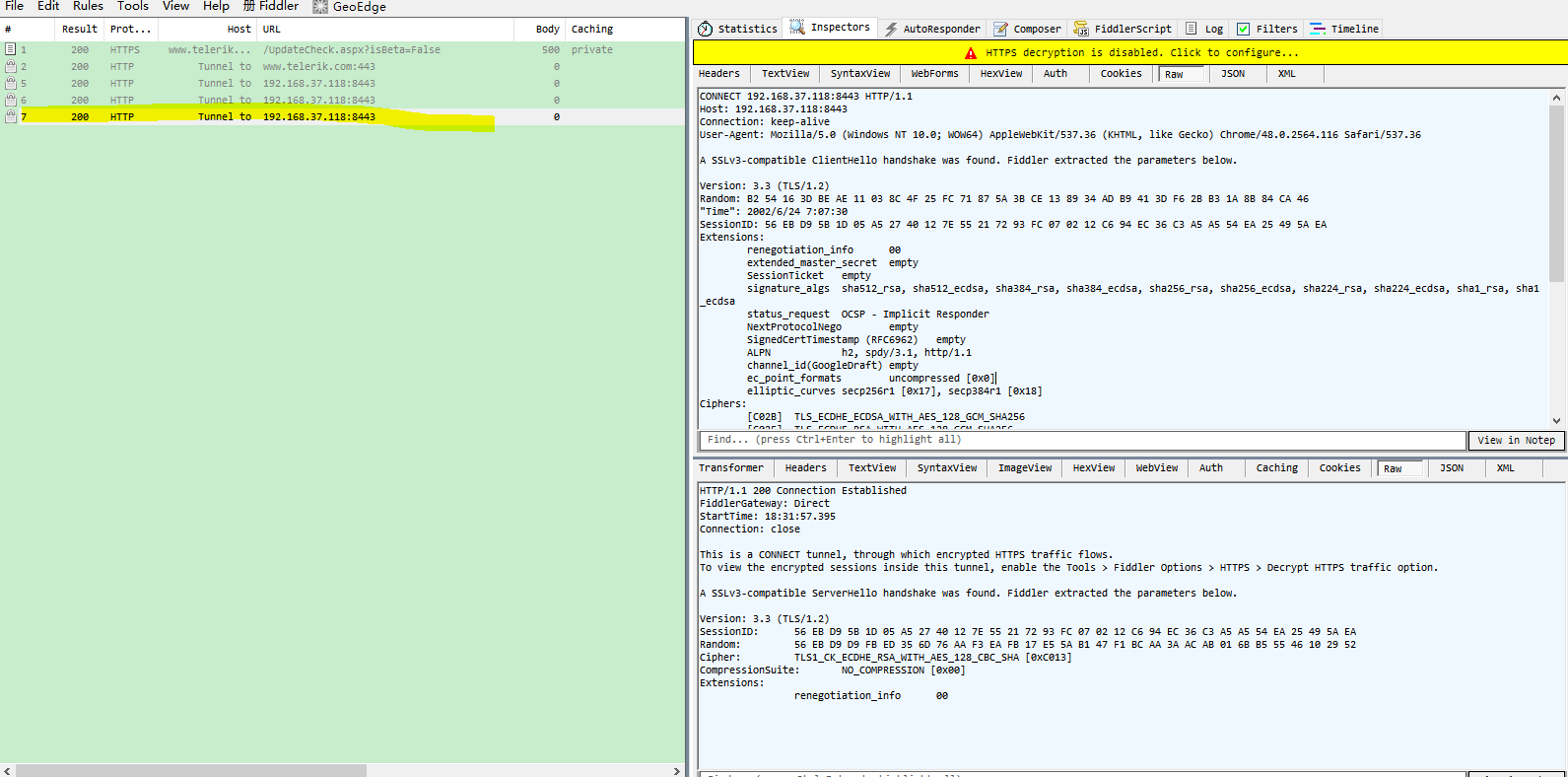
在浏览器中输入`https://192.168.37.118:8443/mobile/school/countries`会有如下提示，点击\*确定\*，然后点击\*继续前往192.168.37.118（不安全）\*即可。



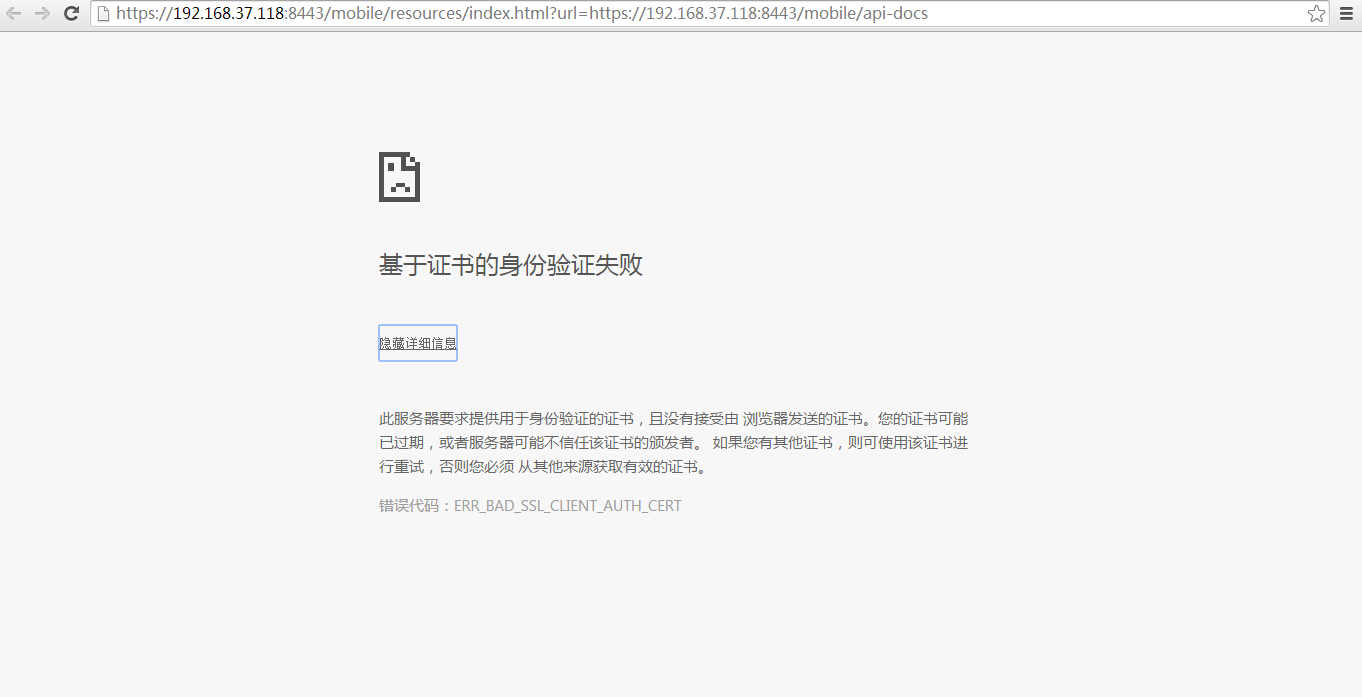




用Fiddler 抓包无法看到明文内容了。



如果没有证书将会有如下提示:



## ####使用HttpClient访问https网站####

参考下面这个例子

[使用HttpClient访问https网站的小例子](https://github.com/toulezu/play/tree/master/testHttpsClient)

## ####参考####

[图解https](http://www.cnblogs.com/zhuqil/archive/2012/07/23/2604572.html)

[Java Tomcat SSL 服务端/客户端双向认证（一）]( <http://www.blogjava.net/icewee/archive/2012/06/04/379947.html>)

[Java Tomcat SSL 服务端/客户端双向认证のApache HttpClient（二）]( <http://www.blogjava.net/icewee/archive/2012/06/05/379983.html>)

[HTTPS的工作原理](http://www.cnblogs.com/ttltry-air/archive/2012/08/20/2647898.html)