SSL

# SSL

SSL 是 Secure socket layer(安全套接层)的缩写。用于网络数据加密传输以及服务器身份验证SSL置身于网络结构体系的传输层和应用层之间。说的再直白点，就是加密解密传输内容的。

SSL 安全协议最初是由美国网景 Netscape Communication 公司设计开发的，全称为：安全套接层协议 (Secure Sockets Layer) ， 它指定了在应用程序协议 ( 如 HTTP 、 Telnet 、 FTP) 和 TCP/IP 之间提供数据安全性分层的机制，它是在传输通信协议 (TCP/IP) 上实现的一种安全协议，采用公开密钥技术，它为 TCP/IP 连接提供数据加密、服务器认证、消息完整性以及可选的客户机认证。

# OSI模型与SSL

OSI（Open System Interconnect），即开放式系统互联；OSI参考模型，是ISO（国际标准化组织）组织在1985年研究的网络互联模型。该体系结构标准定义了网络互连的七层框架（物理层、数据链路层、网络层、传输层、会话层、表示层和应用层），即ISO开放系统互连参考模型。

**(1)**[物理层](http://baike.baidu.com/view/239585.htm)(Physical Layer)

有关传输介质的特性标准，这些规范通常也参考了其他组织制定的标准。连接头、针、针的使用、电流、电流、编码及光调制等都属于各种物理层规范中的内容。物理层常用多个规范完成对所有细节的定义。 协议举例：Rj45，802.3等。

**(2)**[数据链路层](http://baike.baidu.com/view/239592.htm)(Data Link Layer)

它定义了在单个链路上如何传输数据，该层协议与被讨论的各种介质有关。

协议举例：ATM，FDDI等。

**(3)**[网络层](http://baike.baidu.com/view/239600.htm)(Network Layer)

端到端的包传输进行定义，它定义了能够标识所有结点的逻辑地址，还定义了路由实现的方式和学习的方式。为了适应最大传输单元长度小于包长度的传输介质，网络层还定义了如何将一个包分解成更小的包的分段方法。

协议举例：

IP（IPV4、IPV6）internet protocol（网络互联协议）

**(4)**[传输层](http://baike.baidu.com/view/239605.htm)(Transport Layer)

传这层的功能包括是否选择差错恢复协议还是无差错恢复协议，及在同一主机上对不同应用的数据流的输入进行复用，还包括对收到的顺序不对的数据包的重新排序功能。

协议举例：

TCP（transmission control protocol）传输控制协议；

UDP（user datagram protocol）用户数据协议

**(5)**[会话层](http://baike.baidu.com/view/239612.htm)(Session Layer)

会话层主要功能是管理和协调不同主机上各种进程之间的通信（对话），即负责建立、管理和终止应用程序之间的会话。会话层得名的原因是它很类似于两个实体间的会话概念。例如，一个交互的用户会话以登录到计算机开始，以注销结束。

协议举例：无。

**(6)**[表示层](http://baike.baidu.com/view/239615.htm)(Presentation Layer)

表示层处理流经结点的[数据编码](http://baike.baidu.com/view/575608.htm)的表示方式问题，以保证一个系统应用层发出的信息可被另一系统的应用层读出。如果必要，该层可提供一种标准表示形式，用于将计算机内部的多种[数据表示](http://baike.baidu.com/view/1026250.htm)格式转换成[网络通信](http://baike.baidu.com/view/538641.htm)中采用的标准表示形式。数据压缩和加密也是表示层可提供的转换功能之一。

协议举例：无。

**(7)**[应用层](http://baike.baidu.com/view/239619.htm)(Application Layer)

应用层是OSI参考模型的最高层，是用户与网络的接口。该层通过应用程序来完成网络用户的应用需求，如[文件传输](http://baike.baidu.com/view/543341.htm)、收发电子邮件等。

协议举例：HTTP、HTTPS、FTP、WWW、SMTP、telnet；

SSL协议位于TCP/IP协议与各种应用层协议之间，为数据通讯提供安全支持。

比对网络七层协议，会话层的功能和表示层的功能正好和SSL记录协议和握手协议的功能一样，我想这两层协议就应该分别工作在这上面！

# HTTPS与SSL

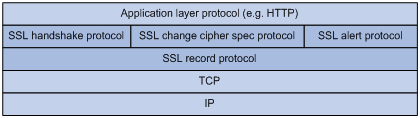
HTTP协议（应用层）是建立在TCP（传输层）协议之上的；

HTTPS协议（应用层）是建立在SSL协议（会话层、表示层）之上的；而SSL协议是建立在TCP（传输层）协议之上的。

我们知道SSL协议本身是干加密解密的，由于SSL是位于传输层和应用层之间的，那么它只能对本层和上层的数据进行加密，管不了下层，所以：HTTPS访问时网址（即请求的网页URL）是加密的，因为URL是HTTP协议数据报头的一部分，而HTTP位于SSL的上层，因此凡是HTTP协议所负责传输的数据就全部被加密了。IP地址是不加密的，因为处理IP地址的协议（IP协议）位于比SSL更低的网络层。

# 协议工作过程

## SSL的分层结构



SSL位于应用层和传输层之间，它可以为任何基于TCP等可靠连接的应用层协议提供安全性保证。SSL协议本身分为两层：

上层为SSL握手协议（SSL handshake protocol）、SSL密码变化协议（SSL change cipher spec protocol）和SSL警告协议（SSL alert protocol）；底层为SSL记录协议（SSL record protocol）。

其中：

SSL握手协议：是SSL协议非常重要的组成部分，用来协商通信过程中使用的加密套件（加密算法、密钥交换算法和MAC算法等）、在服务器和客户端之间安全地交换密钥、实现服务器和客户端的身份验证。

SSL密码变化协议：客户端和服务器端通过密码变化协议通知对端，随后的报文都将使用新协商的加密套件和密钥进行保护和传输。

SSL警告协议：用来向通信对端报告告警信息，消息中包含告警的严重级别和描述。

SSL记录协议：主要负责对上层的数据（SSL握手协议、SSL密码变化协议、SSL警告协议和应用层协议报文）进行分块、计算并添加MAC值、加密，并把处理后的记录块传输给对端。

SSL 协议的版本 1 和 2 只提供服务器认证。版本 3 添加了客户端认证，此认证同时需要客户端和服务器的数字证书。

## SSL握手过程

SSL通过握手过程在客户端和服务器之间协商会话参数，并建立会话。会话包含的主要参数有会话ID、对方的证书、加密套件（密钥交换算法、数据加密 算法和MAC算法等）以及主密钥（master secret）。通过SSL会话传输的数据，都将采用该会话的主密钥和加密套件进行加密、计算MAC等处理。

不同情况下，SSL握手过程存在差异。下面将分别描述以下三种情况下的握手过程：

只验证服务器的SSL握手过程

验证服务器和客户端的SSL握手过程

恢复原有会话的SSL握手过程

### 只验证服务器的SSL握手过程

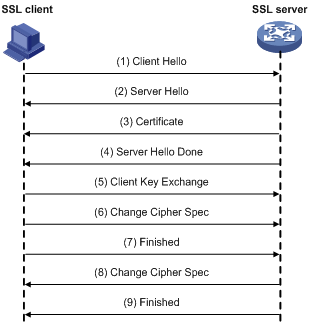


图5 只验证服务器的SSL握手过程

如图5所示，只需要验证SSL服务器身份，不需要验证SSL客户端身份时，SSL的握手过程为：

(1) SSL客户端通过Client Hello消息将它支持的SSL版本、加密算法、密钥交换算法、MAC算法等信息发送给SSL服务器。

(2) SSL服务器确定本次通信采用的SSL版本和加密套件，并通过Server Hello消息通知给SSL客户端。如果SSL服务器允许SSL客户端在以后的通信中重用本次会话，则SSL服务器会为本次会话分配会话ID，并通过 Server Hello消息发送给SSL客户端。

(3) SSL服务器将携带自己公钥信息的数字证书通过Certificate消息发送给SSL客户端。

(4) SSL服务器发送Server Hello Done消息，通知SSL客户端版本和加密套件协商结束，开始进行密钥交换。

(5) SSL客户端验证SSL服务器的证书合法后，利用证书中的公钥加密SSL客户端随机生成的premaster secret，并通过Client Key Exchange消息发送给SSL服务器。

(6) SSL客户端发送Change Cipher Spec消息，通知SSL服务器后续报文将采用协商好的密钥和加密套件进行加密和MAC计算。

(7) SSL客户端计算已交互的握手消息（除Change Cipher Spec消息外所有已交互的消息）的Hash值，利用协商好的密钥和加密套件处理Hash值（计算并添加MAC值、加密等），并通过Finished消息 发送给SSL服务器。SSL服务器利用同样的方法计算已交互的握手消息的Hash值，并与Finished消息的解密结果比较，如果二者相同，且MAC值 验证成功，则证明密钥和加密套件协商成功。

(8)同样地，SSL服务器发送Change Cipher Spec消息，通知SSL客户端后续报文将采用协商好的密钥和加密套件进行加密和MAC计算。

(9) SSL服务器计算已交互的握手消息的Hash值，利用协商好的密钥和加密套件处理Hash值（计算并添加MAC值、加密等），并通过Finished消息 发送给SSL客户端。SSL客户端利用同样的方法计算已交互的握手消息的Hash值，并与Finished消息的解密结果比较，如果二者相同，且MAC值 验证成功，则证明密钥和加密套件协商成功。

SSL客户端接收到SSL服务器发送的Finished消息后，如果解密成功，则可以判断SSL服务器是数字证书的拥有者，即SSL服务器身份验证 成功，因为只有拥有私钥的SSL服务器才能从Client Key Exchange消息中解密得到premaster secret，从而间接地实现了SSL客户端对SSL服务器的身份验证。

&  说明：

Change Cipher Spec消息属于SSL密码变化协议，其他握手过程交互的消息均属于SSL握手协议，统称为SSL握手消息。

计算Hash值，指的是利用Hash算法（MD5或SHA）将任意长度的数据转换为固定长度的数据。

# CA

CA（Certification Authority），是一个证书签证机关 ，该机关管理发放数字证书的。由它发放的证书就叫 CA 证书，以区别于个人使用工具随意生成的数字证书，查看 CA 证书里面有两项重要内容，一个是颂发给谁，另一个是由谁颂发的。

CA 证书的信任链，如果你安装并信任了一个 CA 机构的证书，以他为根，同样也信任了由这个 CA 机构颂发的其它组织的证书，其它组织则可以创建自己的CA中心，颂发下一级的证书，依次往下，一层层会有许多 CA 中心和数字证书。这个最上层的 CA 证书，即可以称为根证书。

[CA证书](https://www.baidu.com/s?wd=CA证书&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3m16YPWc4uWTvP16LnjFh0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtkrH6YPjRLPHb1nWTsrHfYrjR)的种类，CA中心发放的证书分为两类：[SSL证书](https://www.baidu.com/s?wd=SSL证书&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3m16YPWc4uWTvP16LnjFh0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtkrH6YPjRLPHb1nWTsrHfYrjR)和SET证书。一般地说，SSL（安全套接层）证书是服务于银行对企业或企业对企业的电子商务活动的；而SET（安全电子交易）证书则服务于持卡消费、网上购物。虽然它们都是用于识别身份和数字签名的证书，但它们的信任体系完全不同，而且所符合的标准也不一样。简单地说，[SSL证书](https://www.baidu.com/s?wd=SSL证书&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3m16YPWc4uWTvP16LnjFh0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtkrH6YPjRLPHb1nWTsrHfYrjR)的作用是通过[公开密钥](https://www.baidu.com/s?wd=公开密钥&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3m16YPWc4uWTvP16LnjFh0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtkrH6YPjRLPHb1nWTsrHfYrjR)证明持证人的身份。而SET证书的作用则是，通过公开密钥证明持证人在指定银行确实拥有该信用卡账号，同时也证明了持证人的身份。 用户想获得证书时，首先要向CA中心提出申请，说明自己的身份。CA中心在证实用户的身份后，向用户发出相应的数字[安全证书](https://www.baidu.com/s?wd=安全证书&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3m16YPWc4uWTvP16LnjFh0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtkrH6YPjRLPHb1nWTsrHfYrjR)。认证机构发放证书时要遵循一定的原则，如要保证自己发出的证书的序列号各不相同，两个不同的实体所获得的证书的主题内容应该相异，不同主题内容的证书所包含的公开密钥相异等。

# SSL与CA

SSL 协议，是网络通讯的加密协议，用它来加密/解密传输内容，因此它需要一份数字证书，使用里面的公钥加密的内容，或将收到的加密使用私钥将内容解密还原。  
。。 最简单地讲：服务器部署SSL证书后，能确保服务器与浏览器之间的数据传输是加密传输的，是不能在数据传输过程中被篡改和被解密的。所以，所有要求用户在 线填写机密信息的网站都应该使用SSL证书来确保用户输入的信息不会被非法窃取，这不仅是对用户负责的做法，特别是要求填写有关信用卡、储蓄卡、身份证、 以及各种密码等重要信息时，而且也是保护网站自己的以后机密信息的有效手段。而用户也应该有这种意识，在线填写用户自己认为需要保密的信息时看看浏览器右 下面是否出现一个锁样标志，如果没有锁样标志，则表明用户您正在输入的信息有可能在提交到服务器的网络传输过程中被非法窃取而泄露，建议您拒绝在不显示安 全锁的网站上提交任何您认为需要保密的信息，这样才能确保您的机密信息不会被泄露。   
  
。。目前国内用户可选择购买的SSL证书有两种，一种是直接支持所有浏览器的来自美国的Thawte /GeoTrust/Verisign等公司颁发的，一种是国内各种认证中心颁发的，但不被浏览器认可，需要另外安装根证书，同时，在访问网站时会提示 “ [该安全证书由您没有选定信任的公司颁发](http://www.itrus.com.cn/uploadfile/2011/0713/20110713104551256.jpg) ” 或点击锁标志查询证书时会显示“ [无法将这个证书验证到一个受信任的证书颁发机构](http://www.itrus.com.cn/uploadfile/2011/0713/20110713104552675.jpg) ”。用户应该根据自己的需要正确选择全球通用的支持所有浏览器的SSL证书。  
  
。。目前，我国几乎是各个省市都成立了CA(Certification Authority，认证中心)，之所以一窝蜂上CA项目，无非是两个理由：一是商业利益驱动，二是国家安全理由。

# 对称加密和非对称加密

## 对称加密算法

对称加密，或专用密钥（也称做常规加密）由通信双方共享一个秘密密钥。发送方在进行数学运算时使用密钥将明文加密成密文。接受方使用相同的密钥将密文还原成明文。

对称加密是最快速、最简单的一种加密方式，加密（encryption）与解密（decryption）用的是同样的密钥（secret key）。对称加密有很多种算法，由于它效率很高，所以被广泛使用在很多加密协议的核心当中。

对称加密通常使用的是相对较小的密钥，一般小于256 bit。因为密钥越大，加密越强，但加密与解密的过程越慢。如果你只用1 bit来做这个密钥，那黑客们可以先试着用0来解密，不行的话就再用1解；但如果你的密钥有1 MB大，黑客们可能永远也无法破解，但加密和解密的过程要花费很长的时间。密钥的大小既要照顾到安全性，也要照顾到效率。

常用的对称加密算法有：DES、3DES

## 非对称加密算法

非对称加密为数据的加密与解密提供了一个非常安全的方法，它使用了一对密钥，公钥（public key）和私钥（private key）。私钥只能由一方安全保管，不能外泄，而公钥则可以发给任何请求它的人。非对称加密使用这对密钥中的一个进行加密，而解密则需要另一个密钥。比 如，你向银行请求公钥，银行将公钥发给你，你使用银行的公钥对消息加密，那么只有私钥的持有人--银行才能对你的消息解密。与对称加密不同的是，银行不需要将私 钥通过网络发送出去，因此安全性大大提高。

目前最常用的非对称加密算法是RSA算法。

## Md5 是 对称还是非对称？---都不是

对称加密算法，是可逆的（加密（encryption）与解密（decryption）用的是同样的密钥（secret key）），md5加密算法显然不满足；非对称加密更不可能。那它是什么？散列算法或摘要算法。

它通常用于验证消息的完整性。

## 数字信封和数字签名

数字信封：确定接者这得身份。

发送方用一个**随机产生的DES密钥**加密原始数据(消息),然后用接受方的公钥加密DES密钥，称为消息的“数字信封”。将数字信封同DES密钥加密的消息一起发送给接受方，接受者在接收到消息后，先用其私钥对“数字信封”进行解密（打开数字信封），得到发送方的DES密钥，再用此密钥去解密消息。只有用接受方的RSA私钥才能打开数字信封，这样确保了接受者的身份。

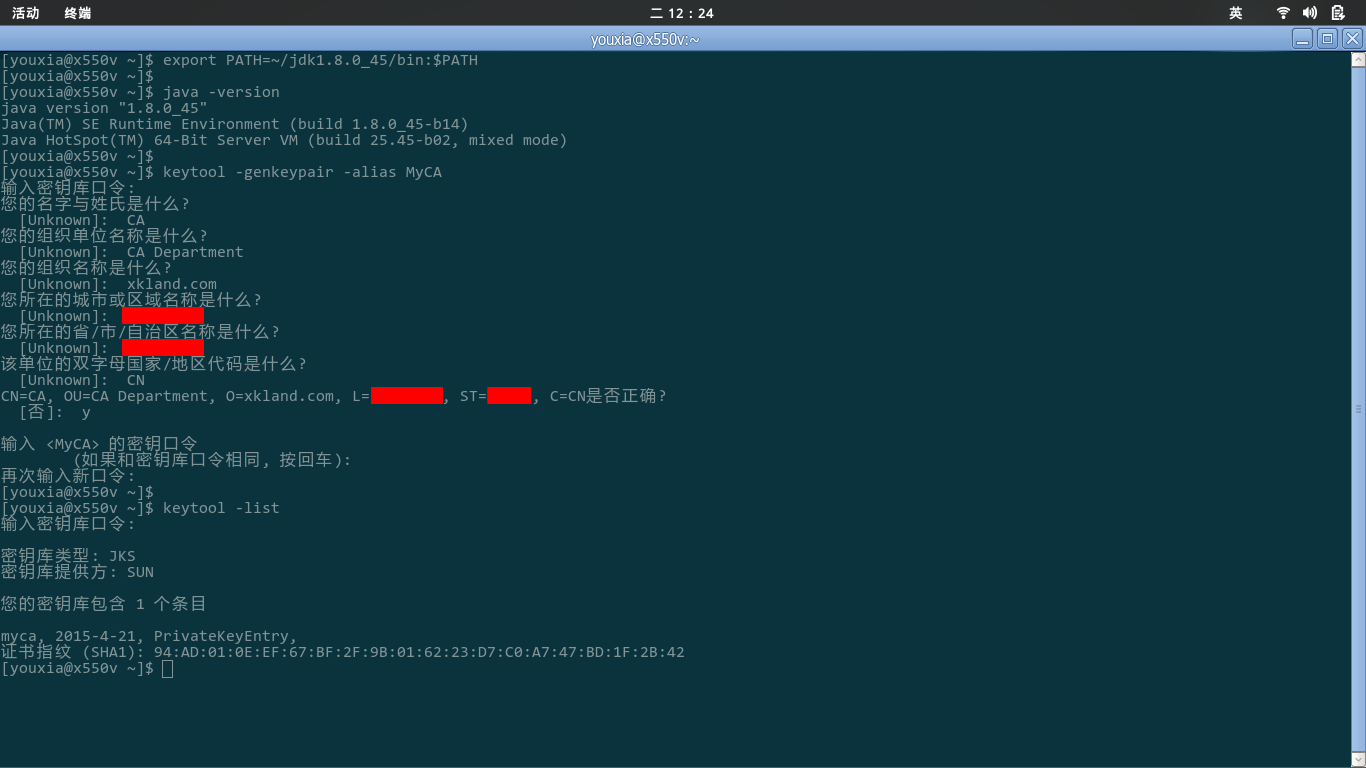
数字签名：确定发送者的身份

消息M用散列函数H得到的消息摘要h1=H(M),然后发送方用自己的双钥密码体制的私钥KSA对这个消息摘要进行加密h’=EKSA(h1)，形成发送方的数字签名.然后这个数字签名将作为消息M的附件随消息一起发送给消息的接收方，消息的接收方首先从接收方的原始消息M中计算出散列函数值h2=H(M),接着用发送双钥密码体制的公钥KPA对数字签名进行解密DKPA(h’)，得h1，如果h1=h2，则表明接收方收到的消息确实是发送方发送的，而且还可以确定此消息没有被修改过.

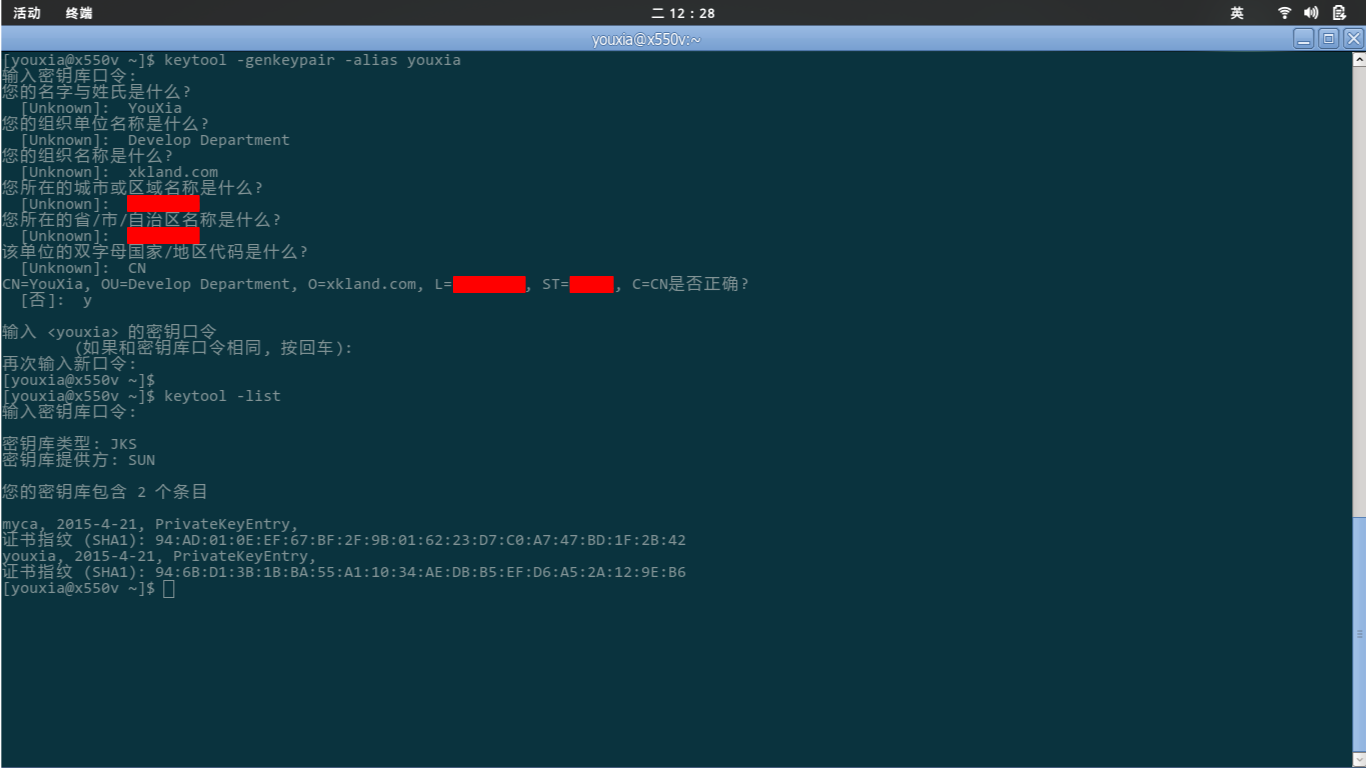
# JDK 生成数字证书

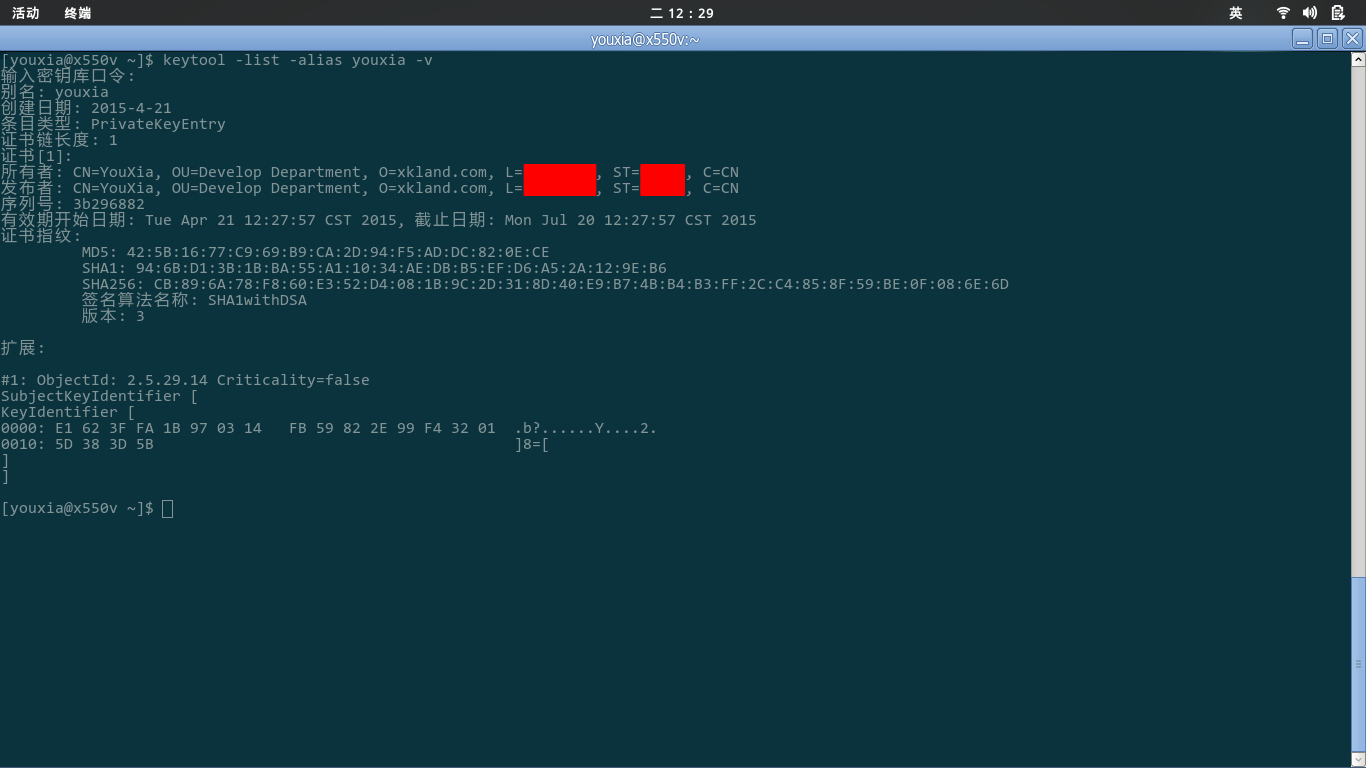
JDK 中创建和管理证书的工具是 keytool 。 keytool 是一个功能强大的安全工具，它不仅仅只是用来创建和管理证书，还可以用来创建和管理对称性加密算法需要用到的密钥，还可以用自己的证书给别人签发证书（类似 CA 的工作），还可以导入别人发布的证书。 keytool 使用 keystore 存储密钥和证书，在一个 keystore 中可以存储多个条目，访问 keystore 和访问 keystore 中的条目均需要密码。

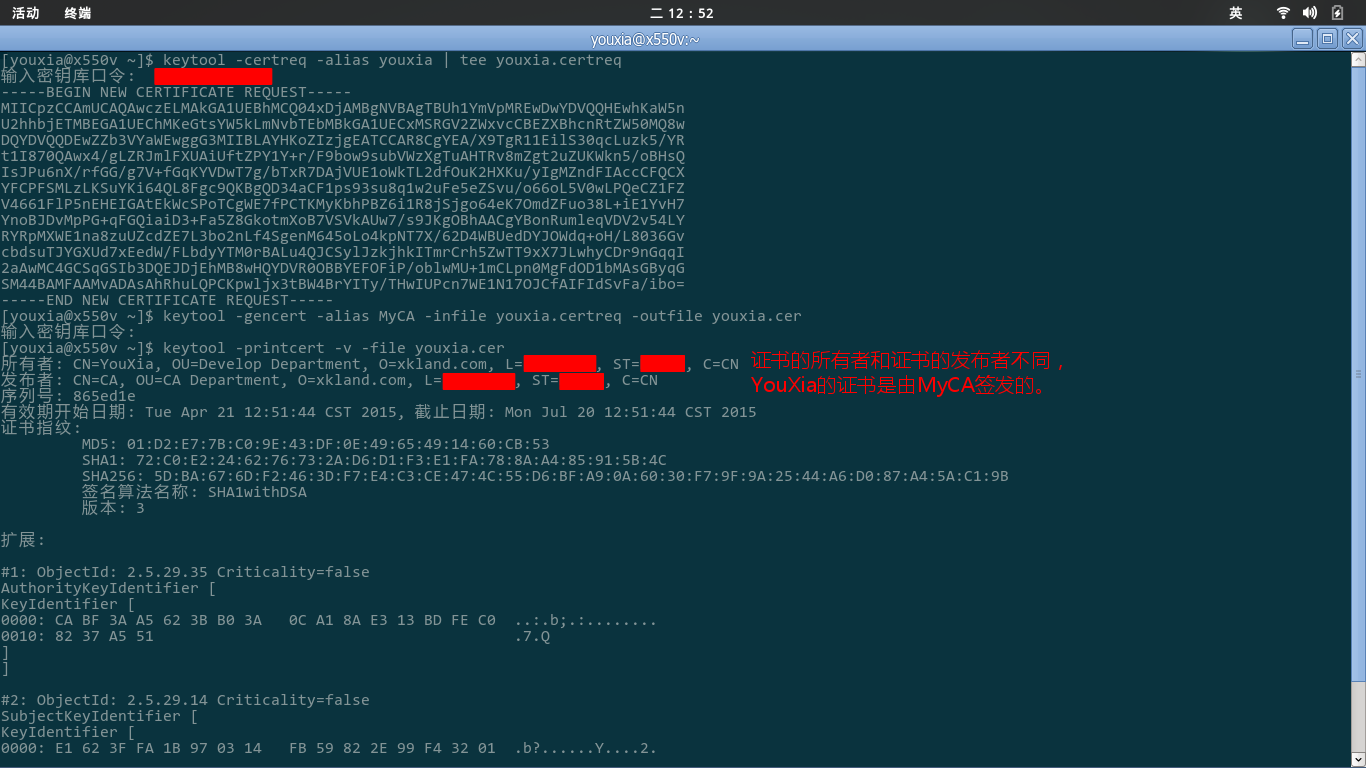
　　下面开始实战。要进行非对称性加密，我们首先要有属于自己的私钥/公钥对，这可以通过 keytool -genkeypairs -alias xxx 命令得到。创建密钥对的时候， keytool 会在 keystore 中生成一个新的条目， -alias xxx 选项就是对该条目进行命名。生成密钥对之后，私钥是以原始数据直接储存在 keystore 中的，而公钥是要发布出去的，所以它被封装在一个 X.509 格式的自签名证书中。换句话说，创建密钥对的时候，同时就创建了一个自签名的证书。

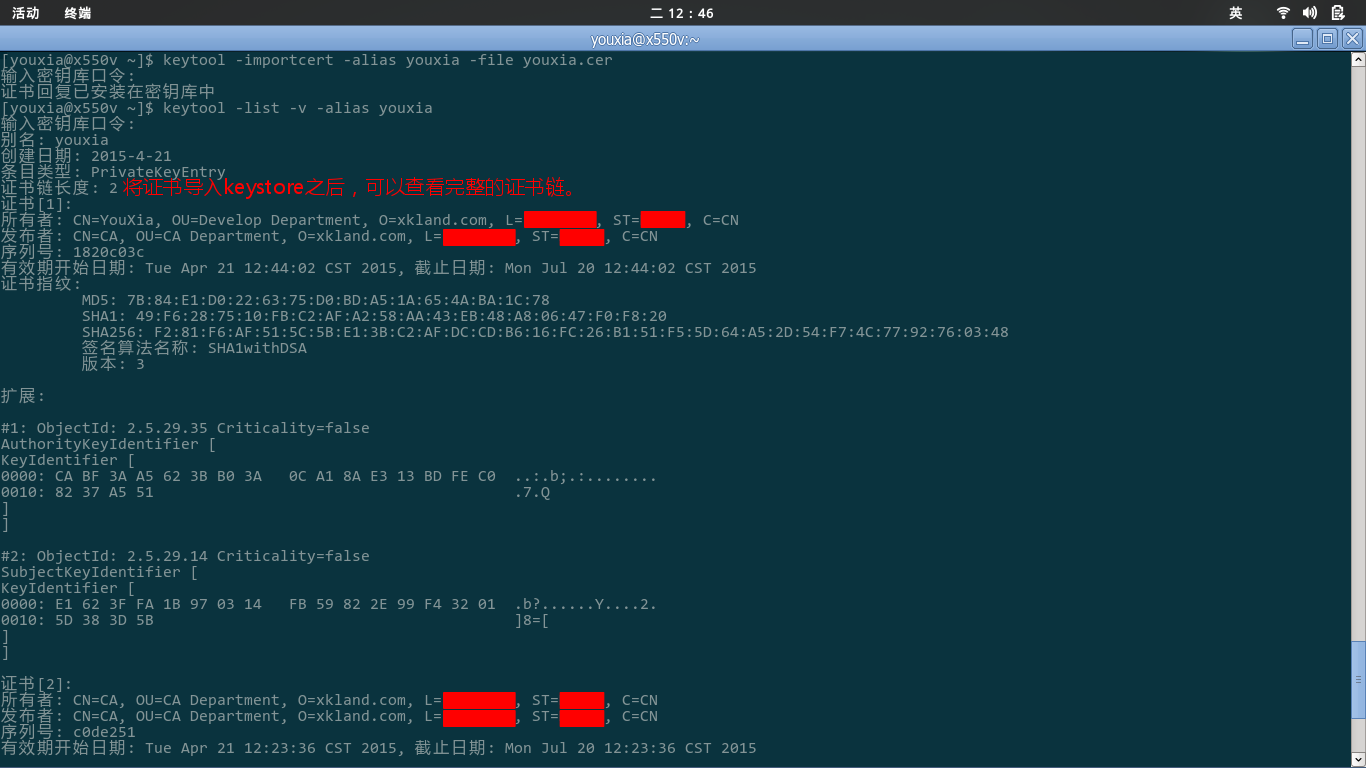
　　先将自己假想为一个认证机构，或者说一个只对我自己签发证书的私有认证机构，我称之为 MyCA，先为 MyCA 生成一个自签名的根证书，使用的命令是 keytool -genkeypair -alias MyCA ，如下图：  
  
　 　可以看到，除了提示我们输入 keystore 的密码和 MyCA 条目的密码之外，还提示我们输入名字、部门、组织、区域、国家代码等信息，这些信息主要是用来标识证书的所有者。按提示完成操作后，就在 keystore 中生成了一个 MyCA 条目，该条目中保存有 MyCA 的私钥和 MyCA 的自签名证书，该证书中包含 MyCA 的公钥。使用 keytool -list 命令看一下，可以看到 keystore 中新增的 MyCA 条目。

　　如果给 keytool -list 命令增加 -v 选项，则可以看到更加详细的内容。这些内容可以让我们对 MyCA 的证书有更直观的了解，如下图：

　　然后，再为我自己生成一个密钥对，使用命令 keytool -genkeypair -alias youxia 命令，按提示完成操作后，keystore 中就又多了一个 youxia 条目，如下图：  


　　使用 keytool -list -v -alias youxia 命令可以查看 youxia 的证书，它也是自签名的，如下图：  


　　自签名的证书可以使用，但是总不如认证机构颁发的证书权威。怎么样让 CA 为我们颁发证书呢？首先我们要向 CA 提交申请，提交申请的时候需要提交一份称为 certificate request 的数据。我们可以通过 keytool -certreq 命令针对 keystore 中相应的条目生成该数据。在这里，我想让 CA 给 youxia 颁发证书，则使用 keytool -certreq -alias youxia 来生成 certificate request，我同时使用管道和 tee 命令让生成的数据既显示在控制台中，又保存在文件 youxia.certreq 中，如下图：  
  
　 　从上图中可以看到，生成的 certificate request 数据是 Base64 编码的。然后，将该申请提交给 CA。当然，现实中的 CA 那都是要收费的，而且还不便宜。那么我只好自己提交给自己了，使用私有的 MyCA 来给 youxia 颁发证书。使用的命令为 keytool -gencert -alias MyCA -infile youxia.certreq -outfile youxia.cer，文件 youxia.cer 就是由 MyCA 颁发的证书。使用 keytool -printcert -v -file youxia.cer 命令可以查看该证书。

　　然后，申请者收到 CA 颁发的证书后，可以使用 keytool -importcert -alias youxia -file youxia.cer 将证书导入到 keystore 中。导入证书后，再使用 keytool -list -v -alias youxia 查看，会看到完整的证书链，如下图：  


　　作为管理工具，keytool 当然提供导出证书的功能，使用 keytool -exportcert 命令即可。至此，我们已经完全了解了生成密钥对和证书的完整流程。关于 keytool 更详细的参数，请大家自己查看 keytool 的文档吧。

## 总结

　　keytool 的使用方法很简单，重要的也就是 keytool -genkeypair、keytool -gencert、keytool -certreq 等命令，看到这些命令基本上就可以想到它们的功能。让人稍稍有点疑惑的是， keytool -genkeypair 不仅仅是生成密钥对，它会同时对公钥进行包装生成自签名的证书， keytool -gencert 并不是凭空生成证书，而是对 certificate request 进行回复。理解这两点，使用 keytool 就畅通无阻了。

<http://wenku.baidu.com/link?url=tR0iavjLqzbZzEnxb4eMLY7ymkSAnFkSgRoNqZmtRhNqwDaWMdR0Khd8RRtDul7hbMsvmaOYl7a9z5aikOHh2cnHbhdY0cTBgX8syAvBSd>\_

<http://blog.csdn.net/zlxdream815/article/details/8158646>

tomcat

http://blog.csdn.net/xxd851116/article/details/18701731

tomcat配置

# 配置Tomcat服务器

1. <Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
2. maxThreads="150" scheme="https" secure="true"
3. clientAuth="false" sslProtocol="TLS"
4. keystoreFile="D:\\tomcat.keystore" keystorePass="888888"
5. truststoreFile="D:\\tomcat.keystore" truststorePass="888888" />

注意服务端证书名保持一致  
属性说明：  
clientAuth:设置是否双向验证，默认为false，设置为true代表双向验证  
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