

之前讲的是公钥加密 私钥解密 这样 大家都给我传数据 只有我一个人可以解密数据

但是 这个 私钥加密 任何人都可以解密

---- 公钥解密成功 ---- 说明一定是那个发送方

\*\*私钥加密 可以起到这个数据确实是这个人发出来的 但是 没有加密的效果

非对称加密的性能很低 现在想做到这个数据是你写的 但是 没有改过 还要有效率

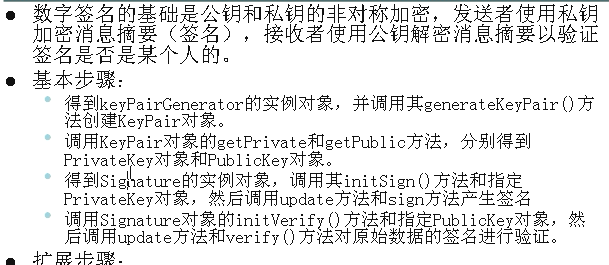
首先 把数据获取MD5 私钥加密MD5 发送出去文档的数据+私钥加密的MD5

接受放就是 对私钥加密的MD5进行公钥解密 --- 解密成功 说明是你发送的 ---- 得到MD5

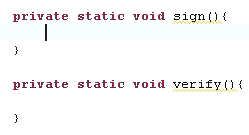
然后对明文文档获取MD5 一样 说明 数据是完整的

现在对数字签名进行编程

签名的单词是Signature



接着前面的代码

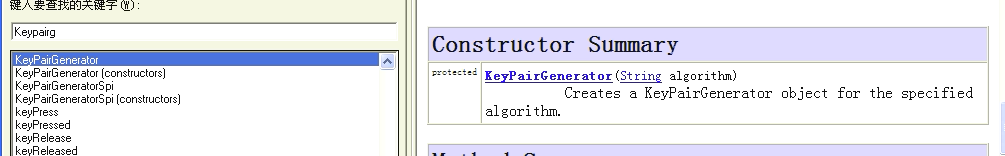


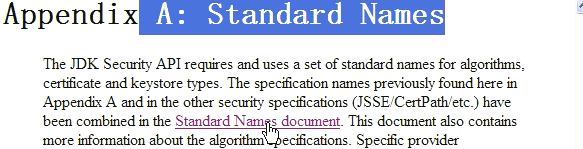
签名也要公钥和私钥对 --- 私钥加密 公钥解密

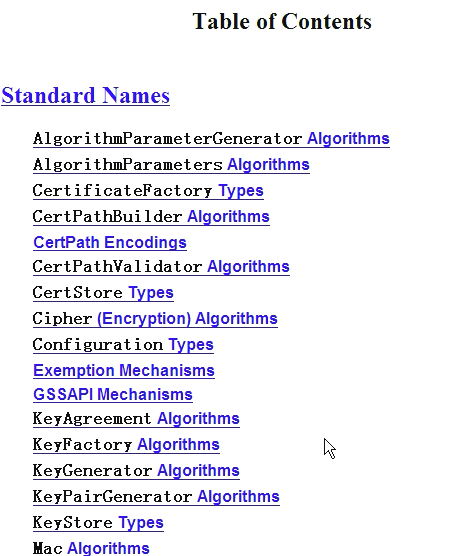
首先获取一对钥匙KeyPair

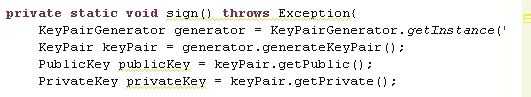


之前学过一个KeyGenerator









现在要使用私钥进行签名

签名的对象是Signature



这个Signature是一个抽象类【默认的实现就是上面的 com.sun.org.apache.xxx就是sun公司的实现类】

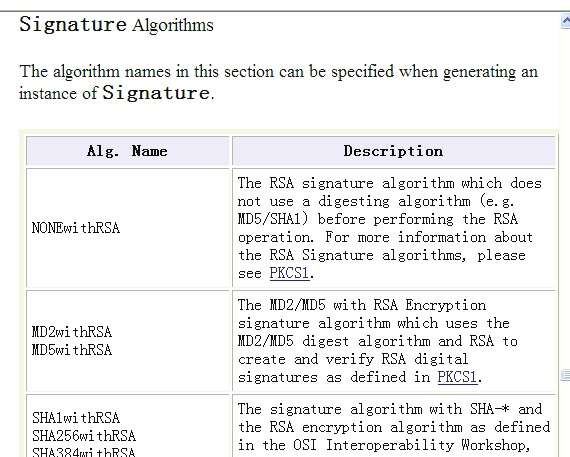
面向接口编程

这个不能直接new

【注意 工厂的单词 XXXFactory 要么是XxxxGenerator 找不到Factory 就找Generator 一般都是通过工厂方法的getInstance()来获取工厂的实例

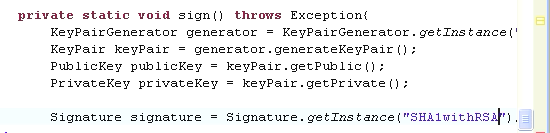
对于有些抽象类 他的对象 可以直接抽象类的getInstance()获取】



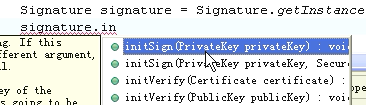


所以 摘要不仅仅是MD5 还有SHA

所以 这里面MD5WithRSA

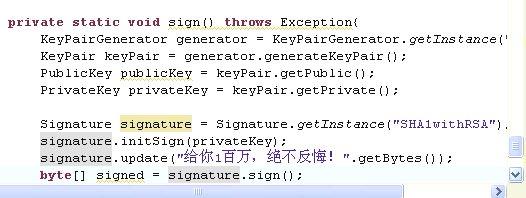


我们这里面使用SHA作为数字摘要 然后 RSA算法进行私钥加密



这里面进行初始化签名 把私钥传递进去

签名结果就是摘要的RSA私钥加密的结果



最后就签名的结果给你 【字符串数据 在我们的模拟中直接复制过去】

然后我们把publickey和 签名的结果 存储起来



------- 之后 我们就做校验 实际上是接收方进行verify 验证

要对明文的数据 计算md5 所以 要把明文数据传递到签名对象进行md5的计算

然后 我们需要发送方的公钥 来解密传递过来的md5

如果解密成功 说明 是发送方发的

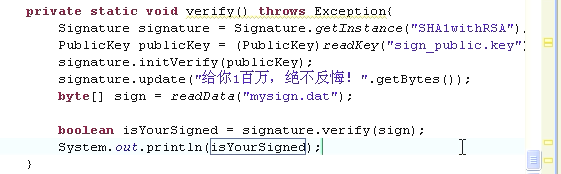
否则 身份验证失败

解密出来 对我们自己本地计算的md5和对方传递过来的md5进行对比

如果md5对比成功 说明数据是完整的

否则 数据完整性校验失败

因此对于封装好的Signature对象 我们就要传入公钥和签名之后的数据



现在调用这两个方法进行测试

 这个就是你签名的

现在另外一个人签了名

运行sign() 把saveKey注释掉 这样产生了一对新的公钥和私钥对 然后 新的签名是新的私钥签出来的

但是 验证方直接读取的是原来的公钥

这样 公钥解密一定出错 所以 会返回false

重新运行 

====== 如果某一个人被冒充了 【马哥举出来过这个例子】 需要引入第三方来公正 保证你的私钥和公钥是谁发放的

谁来认可这个公钥和私钥是你的？

就使用证书 证书也是一种文件格式

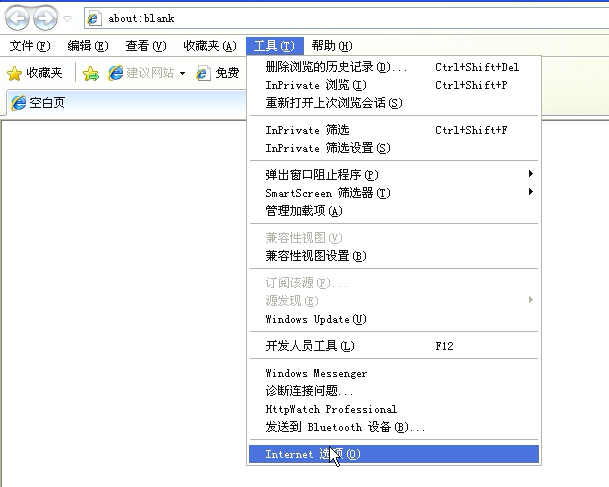
--- 公钥往往装在一个数字证书的载体里面【这个证书 很形象的比喻 就是 盖上了公安局的章 人家一到公安局去查 就能查到 证明这里面的公钥是可靠的公钥】

这个数字证书有一个权威机构的签名 保证这个发送的数字证书就是发送方发的 ---- 这就是数字证书的作用

数字证书

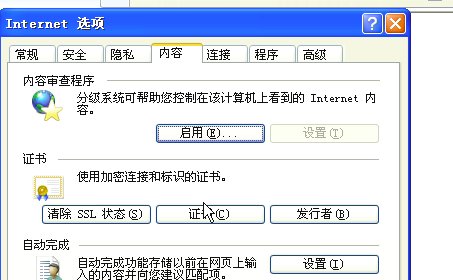
凭什么说你公布的公钥是你的？万一拿了别人的公钥逗人家玩

有几个权威的厂商 可以生产证书

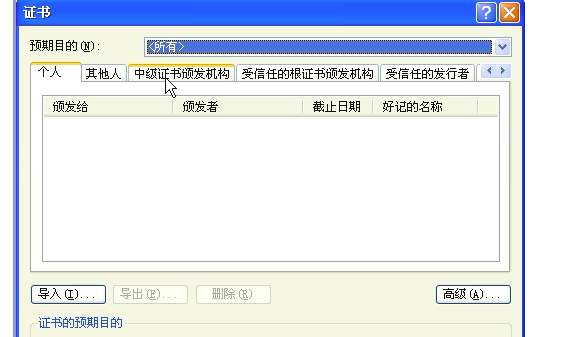


打开之后

在内容选项卡中 有证书这个按钮



点击打开:



点击 受信任的根证书颁发机构



这样信任一批机构 --- 这些好多都是操作系统信任的证书

已经把厂商的证书就安装好了

如果不知名的证书 我的电脑是不认可的

但是 可以到这里面点击某一个厂商 要求他帮助签一个名 说 这个就是我的证书



把我自己产生的公钥 私钥 发送给你选中的机构

然后 人家就调查 最后给你的公钥 私钥签名 就有了证书

厂商就需要相当的严肃

现在要搞一个证书 就要找这个厂商

现在用公安部进行签名 人家一看 就去公安部是不是你签的 人家承认了 但是 又问？你是公安部么？人家说是么？公安部就直接把公安部的证书拿出来 ---然后 用户就找到顶级厂商 问 这个是公安部的证书么？顶级厂商说是 这样就确认了

--- 也就是 我们要前一个证书 不用非要找顶级厂商来签 可以找顶级厂商认定的一些部门进行签名 ---- 也就是找代理进行签名

这就是证书

还有 有的人也给你搞一些证书



自己发送的证书 没有被信任

我们不去网上搞真正的证书

自产自销

在Java中有一个工具 叫KeyTool就会产生证书



KeyTool产生的证书要存储到一个磁盘目录下面 这个目录就称为KeyStore

表示Key的存放地

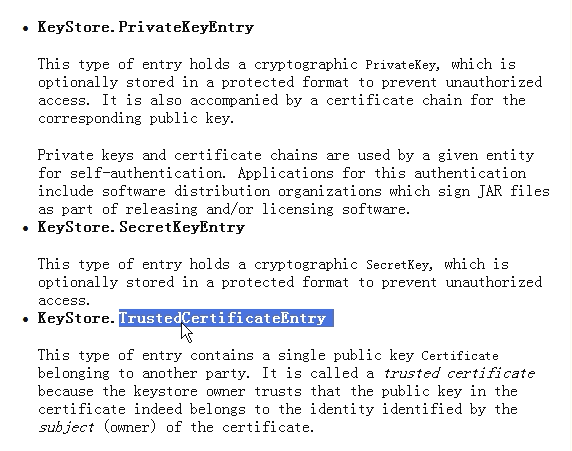
用KeyTool就可以产生这个证书 就可以管理KeyStore

管理就是增删改查

【注意 前面说了 证书实际上就是用来存放公钥的 所以 这个就叫KeyTool 没有叫CertificateTool什么的】

KeyStore里面存储的东西有三类

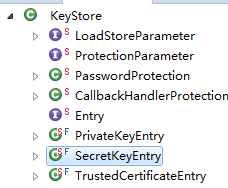




查看了源码

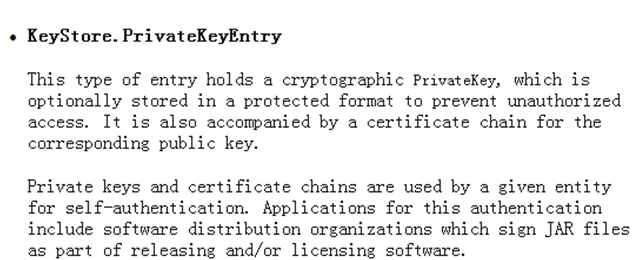
KeyStore.PrivateKeyEntry / KeyStore.SecretKey / KeyStore.TrustedCertificateEntry

【这三个类都是KeyStore下面的静态类成员



第一个KeyStore.PrivateKeyEntry的含义

--- 一个是私钥 – 伴随有一个公钥 这个私钥公钥就要存放到电脑的一个地方

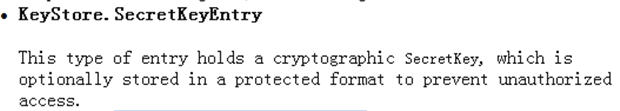


Cryptographic ---- 暗号的 用密码写过的

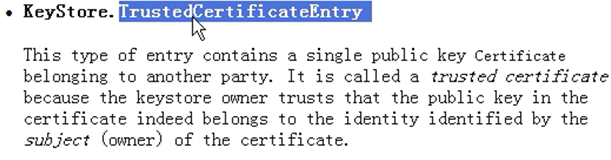
Optionally ---- 随意地

这种类型的entry持有一个私钥PrivateKey ---- 这个PrivateKey随意存储到一个受保护的格式以防止没有认证的访问。这个entry同样伴随一个对应的公钥的证书链。

私钥和证书链用于对一个给定的实体进行自身认证。这样认证的应用包括 对jar文件进行签名以作为发行 和/或者 授权软件 的软件分发组织



这种类型的entry持有一个密钥（对称加密） ----- 后面同上



这种entry包含一个单一属于第三方的公钥证书 --- 因为这个key store的所有者信任那个证书（这个证书确实是属于被证书的所有者验证的实体）里面的公钥，所以这个叫做受信任的证书 ----- 所以 KeyStore存储的是私钥公钥 密钥 和 受信任的证书】

出示了一个证书 我可以把你的证书的复印件存储到我的keyStore中 表示我信任证书 下次再来 我就到我的keyStore中找 找到了 就不用到公安部进行校验了

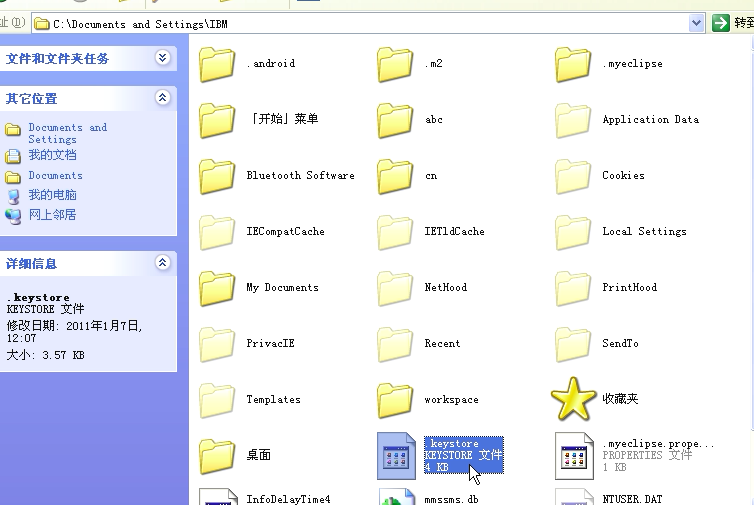
每次校验 信任了 就把你的证书存储在我家的抽屉里面 虽然不是我自己的证书

所以KeyStore中可以存很多东西

现在 我们要存储私钥 和 私钥配对的公钥 但是 公钥要存储在证书里面 所以 这里面存储的是 ---- 私钥 和 对应的存储公钥的证书

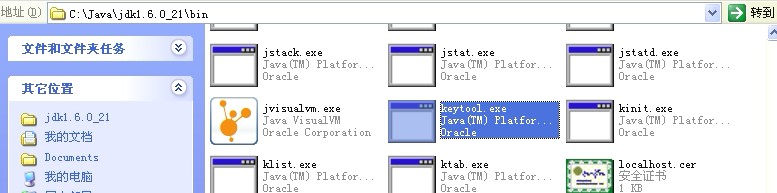
但是 这个证书刚刚没有盖章 也就是没有签名

现在 我们产生证书 默认存放在：

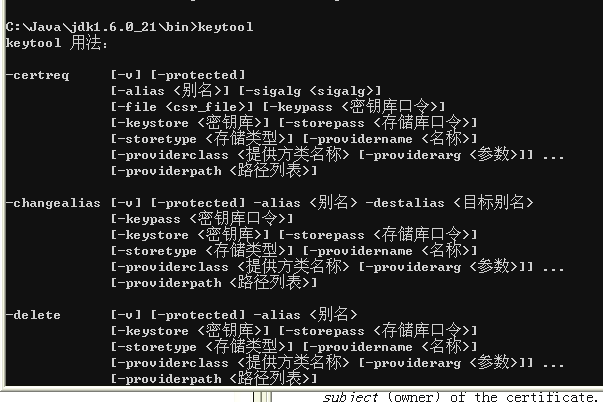


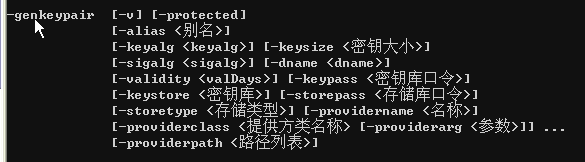
现在删除.keystore 这样我的电脑就没有这个证书了

 jdk安装目录的Bin下面有一个keytool.exe



我们在命令行中使用一下这个工具

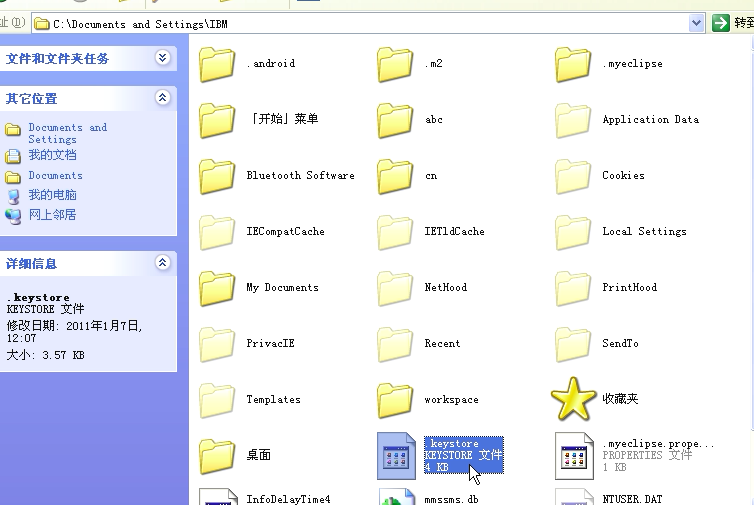




-genkeypair 就是可以产生公钥私钥对

每一个证书都应该有一个别名 用于区分多个名字

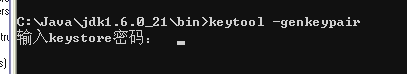
 这个默认的位置



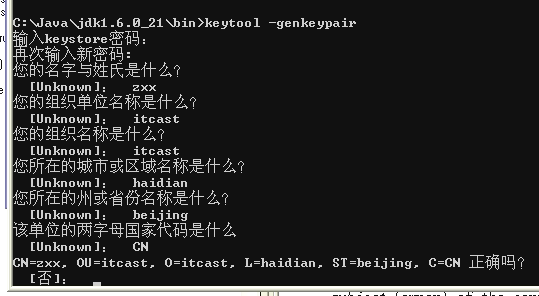
如果想对证书加上密码 也就是 你要看我的证书 那么你就要输入密码才行 所以 对于证书也可以进行加密

所以有一个-keypass选项

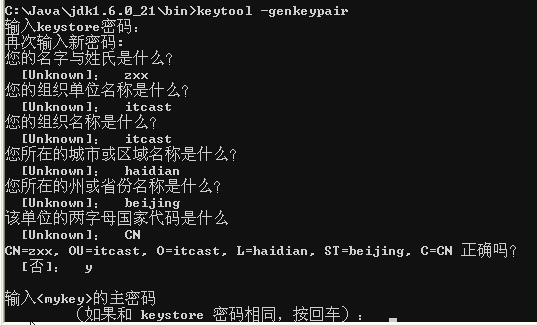
现在 使用一下这个工具



第一次没有 输入一个初始化密码



直接回车就是否 想确认 直接使用汉字是 或者y



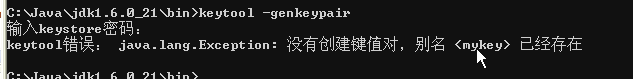
回车



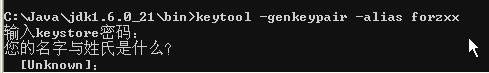
这样就生成了 放了 证书 私钥

接下来 刚才key有个别名

刚才生成了一个证书 别名默认就是mykey ----- 再来产生一个证书

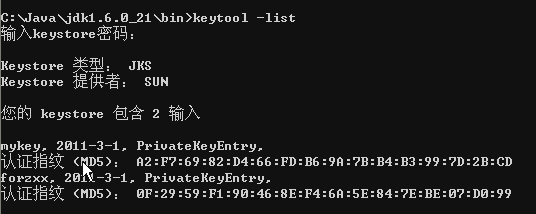


所以 报错告诉我们 我们要为证书重新指定一个别名

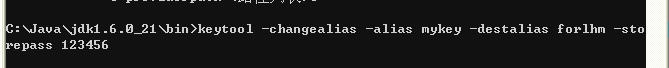


这样就不报错了

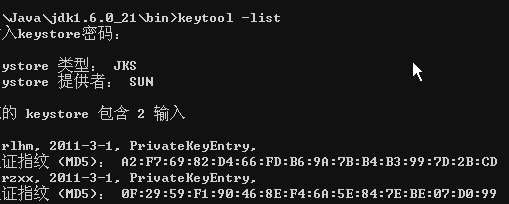
查看所有的证书：



觉得forzxx太难听 进行更换别名



这样就更改完了



显示更加详细的内容

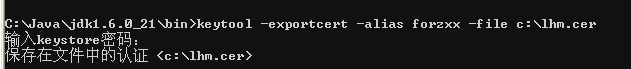




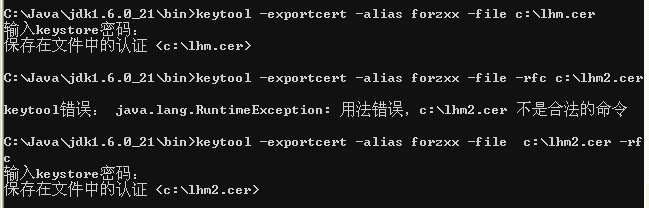


现在到出一个证书

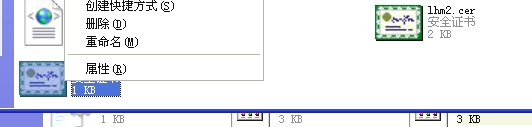




再次到出一个证书

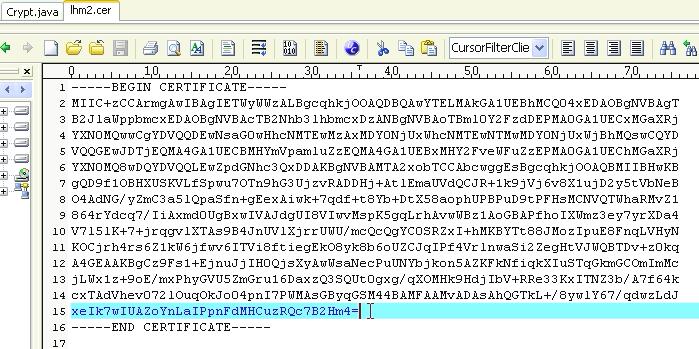


到C盘下面看看

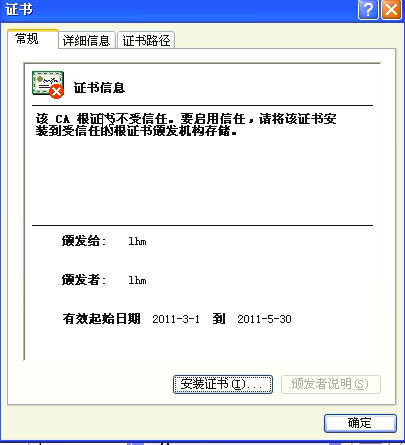
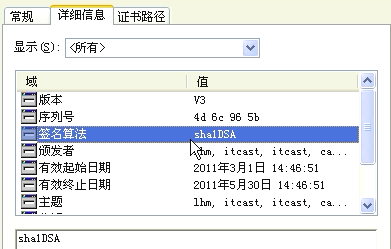


打开之后看不懂 不是文本的

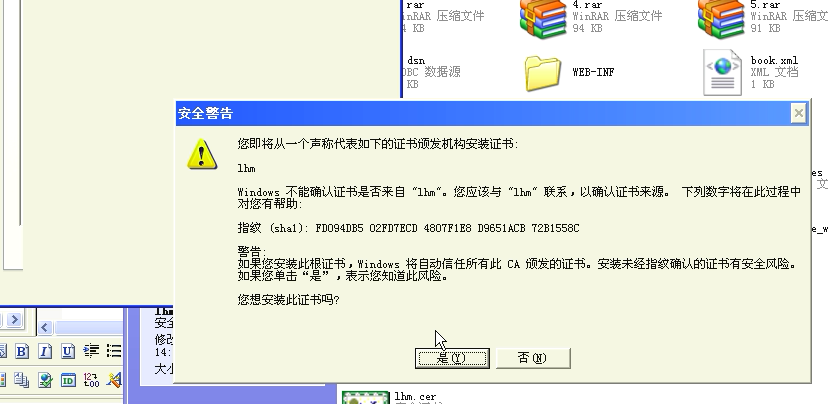
打开第二个



我的证书刚才打印出来都是二进制的 很多字符打不出来 使用编码变成文本 文本的形式可以打印出来

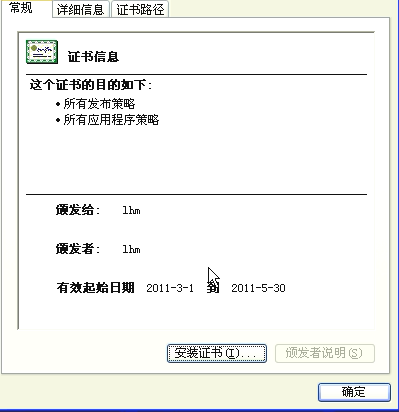
 

发给你 你点击安装证书



信任之后

再打开就不报错了





导入证书

先删除证书



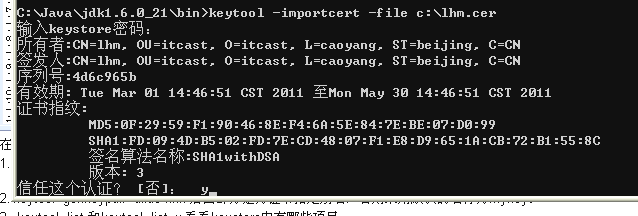
删除完成了



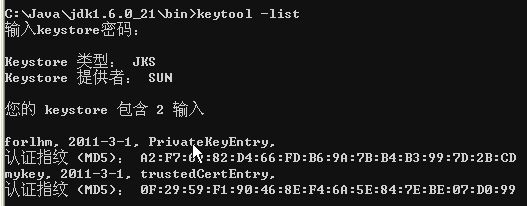
就剩一个了

现在import这个证书





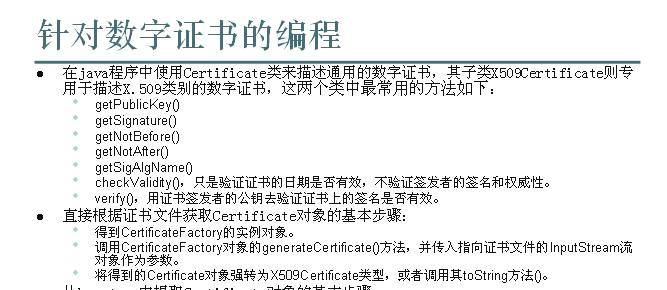
接下来



证书上面不带私钥 后面导引来的

导入别人的证书何在？以后你的电脑就信任这个证书 以及信任这个证书签发的其他的证书。

======针对证书进行编程



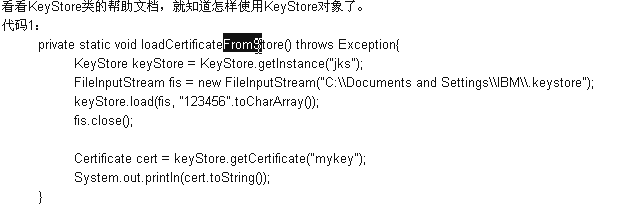
证书在磁盘上面是文件的形式 但是 在Java中是有一个类来表示这个证书的



证书的来源有几个地方：直接从KeyStore中读出来的

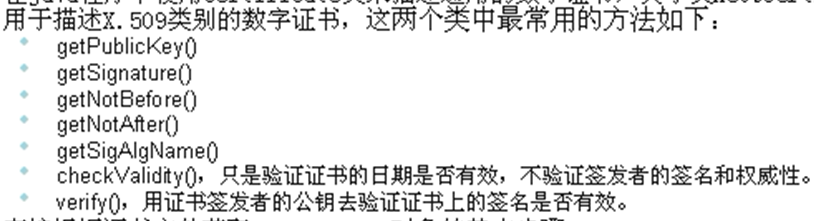
还有就是给你的文件 非要编程Java内存的对象

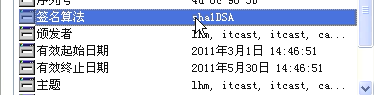
从KeyStore中装载文件



这个和-list –v的效果一样

证书上面有什么？



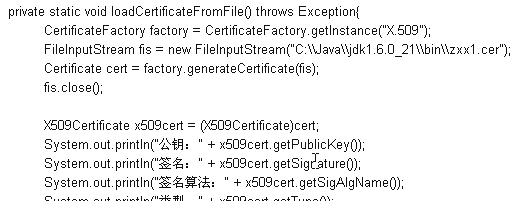


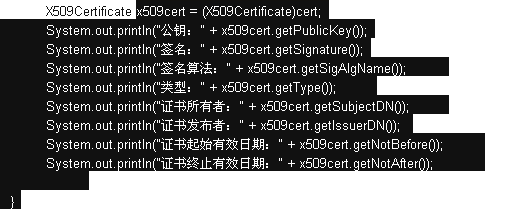
还要校验这个证书的日期是否有效



这个jks是Java key store的意思

还有一个是给了一个文件 要得到一个证书：





这个证书也是有格式的 证书的格式是给电脑看 不是给人看的 这个证书的格式 是X.509

现在的证书 格式都是X509