JavaSE6基于JSR105的XML签名之实践篇 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/252/2021_2022_JavaSE6_E5_9F_BA_c104_252500.htm 一、 密码学密钥和证书 现在，我们已经准备好我们的XML签名示例应用程序。 让我们首先分析下列XML文档-./etc/invoice.xml：＜?XML version="1.0" encoding="UTF-8" standalone="no"?＞＜invoice XMLns="http://www.company.com/accounting"＞＜items＞＜item＞ ＜desc＞Applied Cryptography＜/desc＞ ＜type＞book＜/type＞ ＜unitprice＞44.50＜/unitprice＞ ＜quantity＞1＜/quantity＞ ＜/item＞＜/items＞＜creditcard＞ ＜number＞123456789＜/number＞ ＜expiry＞10/20/2009＜/expiry＞ ＜lastname＞John＜/lastname＞ ＜firstname＞Smith＜/firstname＞＜/creditcard＞＜/invoice＞ 我们计划使用一个XML签名对它进行签名并且希望使用一个基于一个公共密钥的签名方法。 让我们先生成密码学密钥。为此，我们可以使用JDK中提供的keytool工具-把该程序移动到./etc文件夹下，并且执行下列命令：keytool -genkey -keysize 512 -sigalg DSA -dname "cn=Young Yang， ou=Architecture， o=Company， L=New York， ST=NY， c=US" -alias biz -keypass kp1234 -keystore bizkeystore -storepass sp1234 -validity 180 这个命令能够创建密钥并预以存储-名字为bizkeystore，存储在工作目录./etc下，并且指定它的口令为sp1234。它还生成一个针对实体（它包含有一个卓著的名字-Young Yang）的公有/私有密钥对。【注意】，这里使用DSA密钥生成算法来创建公有/私有密钥-都为512位长。上面的命令进一步创建了一个自签名的证书，这是使

用SHA1的DSA算法(JSR-105注释中的DSA_SHA1，其中包括了公共密钥和前面那个卓著名字信息)实现的。这个证书将保持180天的有效期并且关联与一个密钥存储文件（此处引用的别名为"biz"）中的私有密钥。该私有密钥被赋予口令kp1234。 我们的示例中包括一个简单的Java类-KeyStoreInfo，用于把存储于前面的密钥存储文件中的密钥和证书信息输出到System.out；这个类也用于应用程序从中取得密钥对-这里的私有和公共密钥匹配作为输入参数指定的条件。为了试验它能够输出包含在前面存储文件bizkeystore中的信息，读者可以运行Ant目标ksInfo。 下列代码片断显示KeyStoreInfo中的用来检索一个KeyPair的方法：public static KeyPair getKeyPair(String store，String sPass，String kPass，String alias)throws CertificateException，IOException，UnrecoverableKeyException，KeyStoreException，NoSuchAlgorithmException{ KeyStore ks = loadKeyStore(store，sPass). KeyPair keyPair = null. Key key = null. PublicKey publicKey = null. PrivateKey privateKey = null. if (ks.containsAlias(alias)){ key = ks.getKey(alias，kPass.toCharArray()). if (key instanceof PrivateKey){ Certificate cert = ks.getCertificate(alias). publicKey = cert.getPublicKey(). privateKey = (PrivateKey)key. return new KeyPair(publicKey，privateKey). }else{ return null. } } else { return null. }} 借助于一个KeyPair，我们可以容易地得到PrivateKey和PublicKey-通过调用相应的操作getPrivate()和getPublic()实现。 为了从KeyStore中得到一个PublicKey，我们并不真正需要在上面的方法中所要求的密钥口令，而这正是下列方法所实现的

：public static PublicKey getPublicKey(String store，String sPass，String alias)throws KeyStoreException，NoSuchAlgorithmException，CertificateException，IOException{ KeyStore ks = loadKeyStore(store，sPass). Certificate cert = ks.getCertificate(alias). return cert.getPublicKey().} 在上面两部分代码片断中，方法KeyStore loadKeyStore(String store，String sPass)是一个工具函数，用于实例化一个KeyStore对象，并且从文件系统加载入口。我们以如下方式实现它：private static KeyStore loadKeyStore(String store，String sPass)throws KeyStoreException，NoSuchAlgorithmException，CertificateException，IOException{ KeyStore myKS = KeyStore.getInstance("JKS"). FileInputStream fis = new FileInputStream(store). myKS.load(fis，sPass.toCharArray()). fis.close(). return myKS.} 伴随JDK提供的keytool还可以把存储在一个密钥储存文件内的证书输出到系统文件中。例如，为了创建一个包含X509证书（关联于别名为biz的密钥入口）的biz.cer文件，我们可以从文件夹./etcdirectory下运行下列命令：keytool -export -alias biz -file biz.cer -keystore bizkeystore -storepass sp1234 这个证书实现认证我们讨论上面的公共密钥。 我们还在示例中包括了一个Java类-CertificateInfo，用于把一个证书中的一些有趣的信息输出到System.out。为了试验这一点，读者可以运行Ant目标certInfo。然而，要理解该代码及其输出，必须具有DSA和RSA算法的基本知识。当然，读者可以安全地绕过这个程序而继续阅读本文后面的内容。