核心技术在Applet中实现数字签名Java认证考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/644/2021_2022__E6_A0_B8_E5_BF_83_E6_8A_80_E6_c104_644634.htm 准备工作：当然是写一个Applet,其中涉及到对本地文件的读或写操作用于测试目的。可以先予以执行，想必应该是会抛出java.security.AccessControlException的。 下面实现签名：（代码本身无须任何更改） 注：下面所使用的keytool或jarsigner命令只存在与Java2中，1.1中为javakey.且二者的安全模型有很大的差异，故务必首先确定你所使用SDK的版本。 （1）使用keytool命令建立自己的密钥库并生成X.509自签名证书。 下列的命令很长时间都不使用了，具体参数什么的都忘完了，故只给出大概的参数，至于具体用法请参阅文档。 keytool -genkey -alias xiewuhen(这里似乎应该是密钥库的项目)；下面按照提示一步步进行下去即可。 （2）把Applet类文件打包成jar. （3）利用jarsigner签名jar文件。 jarsigner applet.jar xiewuhen(这个applet.jar是你要签名的jar文件，xiewuhen是你前面在密钥库中生成的项目) （4）检查jar文件，在meta-inf目录下应该会出现两个新文件，xiwehen.SF,xiwuhen.DSA.注意xiewuhen这个密钥项目只是事例，具体是什么依赖你最初使用keytool所生成的项目。其中SF文件是签名文件，是一个ASCII文件，保存了签名过程中使用的摘要算法。DSA文件是一个二进制文件，保存了签名和数字证书。最后一点就是文件扩展名具体是DSA还是RSA取决你使用的签名算法，默认是DSA，可以在keytool参数中指定RSA算法。 （5）若你是在单机且只有一个密钥库的机器上测试，下一个步骤可以省

略因为这牵涉到证书的导入导出的问题。由于你签名方和验证方共用同一个密钥库，那证书就无须导来导去了。但为了模拟真正的环境，所以还是建议在多机上测试或建立多个密钥库。（6）使用keytool -export -file -alias等一系列参数将证书导出，保存成crt格式，可以在流览器中配置该证书。但我们暂不考虑这种情况。再次使用keytool -import -file -alias命令将证书导入验证方的密钥库。注意alias指定的别名务必和生成时的一模一样。（7）修改策略文件。该文件位于主目录的/lib/security/中，是java.seucrity文件。至于你的Java主目录是什么，可以查看java.home系统属性。（8）如下修改策略：//指定的的密钥库路径。type目前只能是JKS(Sun的默认实现)，可以省略。 keystore "keystoreURL","keystoreType". grant signedBy "xiwuhen" codeBase "这里指定Applet的装入URL,若是文件URL须在前面加上file: 注意这里务必不可省略，否则Applet就会共享其他的策略了{ permission java.io.FilePermission "autoexec.bat","read". //上面的权限只是一个例子，依据你的要求给出具体策略。 }. 最后还有一点就是Applet可以有许多人顺序签名，若是那样的话，signedBy中的各个签名者之间是和的关系，而不是或。也就是说比如signedBy "a,b"那必须两人都签了名名才行，只有其一是不予通过的。 关于证书的问题,由于这里我们使用的是自签名证书，所以只能用于测试目的，在真正的商务活动中是不会有人信任这种证书的（想必你也不会信任来路不明的证书）。要想真正用于实际须向证书机构（CA）申请并购买数字证书。类似机构大的主要有Verisign，Thawte等。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com