20180909

阅读前可先阅读Java中安全API内容

# 出口限制

<https://www.oracle.com/technetwork/java/javase/downloads/jce-6-download-429243.html>

将下载完后文件覆盖至JDK或JRE 下lib\securtiy目录

比如AES算法若无该文件则不能使用 256位的加密

KeyGenerator kg = KeyGenerator.getInstance( "AES" );

kg.init( 256 );

SecretKey secretKey = kg.generateKey();

System.out.println( new String( secretKey.getEncoded() ) );

否则将抛出 InvalidKeyException

# 配置加密组件Bouncy Castle

包括许多JCE不提供的算法，如MD4、IDEA、国际数据加密算法。

<!-- https://mvnrepository.com/artifact/org.bouncycastle/bcprov-jdk15on -->

<dependency>

<groupId>org.bouncycastle</groupId>

<artifactId>bcprov-jdk15on</artifactId>

<version>1.59</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.bouncycastle/bcprov-ext-jdk15on -->

<dependency>

<groupId>org.bouncycastle</groupId>

<artifactId>bcprov-ext-jdk15on</artifactId>

<version>1.59</version>

</dependency>

以MD4算法为例：

byte[] data = "Helloworld".getBytes();

MessageDigest md = MessageDigest.getInstance("MD4");

md.update(data);

byte[] digest = md.digest();

System.out.println( new String(digest));

若无配置则抛出MD4 MessageDigest not available异常

## 配置

添加至 lib\security\java.security中

security.provider.11=org.bouncycastle.jce.provider.BouncyCastleProvider

并将 bcprov-ext-jdk15-149.jar放在lib的ext中

## 调用方式

有时候需要显式引入Provider,首先将bcprov-jdk15-159.jar引入工程

则需要在初始化密钥工厂、密钥引擎之前调用如下代码：

Servurity.addProvider( new BouncyCastleProvider() )

或者，在初始化密钥工厂、密钥生成器等引擎类，采用如下方式：

MessageDigest md = MessageDigest.getInstance( “MD4”, “BC” );

## Bouncy Castle简介

JCE工具及其扩展包：仅包括 org.bouncycastle.jce。这是对JCE的支持。

JCE支持者及其测试包：包括org.bouncycastle.jce.provider包及其子包。

轻量级加密包：包括org.bouncycastle.crypto包及其子包。

OCSP和OpenSSL PEM支持包。

ASN.1编码支持包

工具包

其他包、

# 对称加密网络应用

相关可参考算法 对称加密部份

# 数字证书

20180917 可先阅读网络协议HTTPS关于数字证书相关内容

## 基本知识

数字证书又称为电子证书，具备常规加密/解密算法，包含签名算法，*可用于网络数据加密/解密交互，标识网络用户身份。*

VeriSign、GeoTrust、Thawte是国际权威数字证书颁发认证机构的三巨头。其中，应用最为广泛的是VeriSign签发的电子商务用数据证书。

CAcert 是一个免费的数字证书颁发国际组织，随着用户群的增大和颁发手段的可信性，这种免费的数字证书可信度也越来越高。

数字证书采用了公钥基础设施(Public Key Infrastructure,PKI)，使用了相应的加密算法确保网络应用的安全性。

非对称加密算法：对数据进行加密/解密操作，确保数据机密性

数字签名算法：对数据进行签名/验证操作，确保数据的完整性和抗否认性

消息摘要算法：对数字证书本身做摘要出来，确保数字证书的完整性

最常用非对称加密算法：RSA算法

与之配套签名算法：SHA1withRSA算法

最为常用的消息摘要算法：SHA1

除了使用RSA算法外，我们还可以使用DSA算法。只是使用DSA算法无法完成加密/解密实现，即这样的数字证书不包括加密/解密功能

## 数字证书文件编码格式

CER是数字证书的一种编码格式，它是BER的一个变种，比BER的规定更加严格。

DER同样是BER的变种，与CER相比：DER使用定长模式，CER使用变长模式。

所有证书复合公钥基础设施PKI制定的ITU-T X509国际标准，目前已有3个版本。目前金融行业里，比较常见的是X.509V3的证书，也就是遵循X.509第三个版本的标准数字证书。

通常使用Base64编码格式作为数字证书文件存储格式

获得数字证书后，可以将其保存在电脑里，也可以保存在IC卡或USB Key中。

我们使用银行借记卡或信用卡进行网上交易时，为增强网络数据传输安全性使用银行提供的“U盾”或其他设备。这种设备中实际存储了银行提供的数字证书，并通过设备上固化的程序对数字证书进行升级。

## 模型分析

实际应用中，很多数字证书都属于自签发证书，即证书申请者为自己的证书签发。这类证书通常应用于软件厂商内部发放的产品中，或约定于使用该证书的数据交互双方。数字证书完全充当加密算法的载体，为必要数据做加密/解密和签名/验证等操作。

### 证书签发

证书申请者

认证机构

构建密钥对

构建密钥对

发送算法、公钥和申请者部份信息

颁布证书

证书申请者

认证机构

构建密钥对

构建密钥对

发送算法、公钥和申请者部份信息

颁布证书

### 加密交互

客户端

服务端

使用公钥加密

发送数据

使用私钥解密

回应加密数据、数字签名

客户端

服务端

使用私钥对待数据加密

使用私钥对待加密数据签名

使用公钥对数据解密

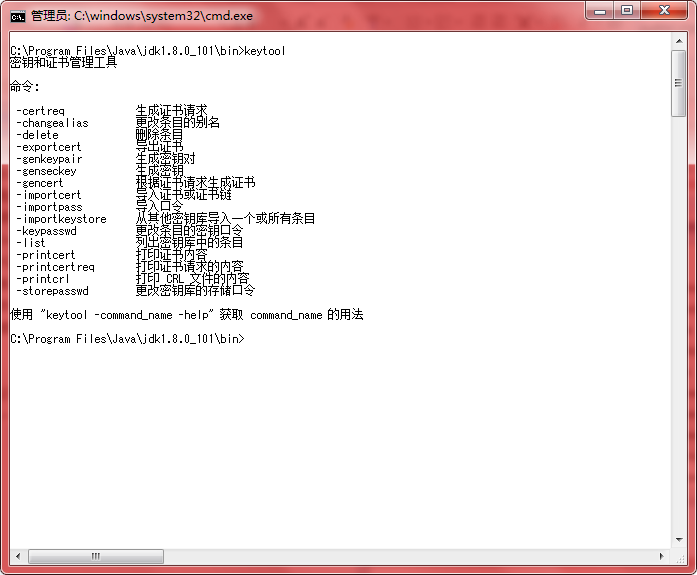
使用公钥和解密数据验证签名

数字证书最佳应用环境是HTTPS安全协议中，使用流程远比上述加密交互流程复杂。但相关操作封装在传输层，对于应用层透明。在HTTPS安全协议中使用非对称加密算法交换密钥，使用对称加密算法对数据进行加密/解密操作，提高加密/解密效率。

## 证书管理

### 使用KeyTool

KeyTool是Java中数字证书管理工具，用于数字证书的申请、导入、导出和撤消等操作。



#### 构建自签名证书

在构建证书之前，需要生成密钥对，即基于某一种非对称加密算法的公私钥。

生成密钥对1

Keytool

-genkeypair

-keyalg RSA

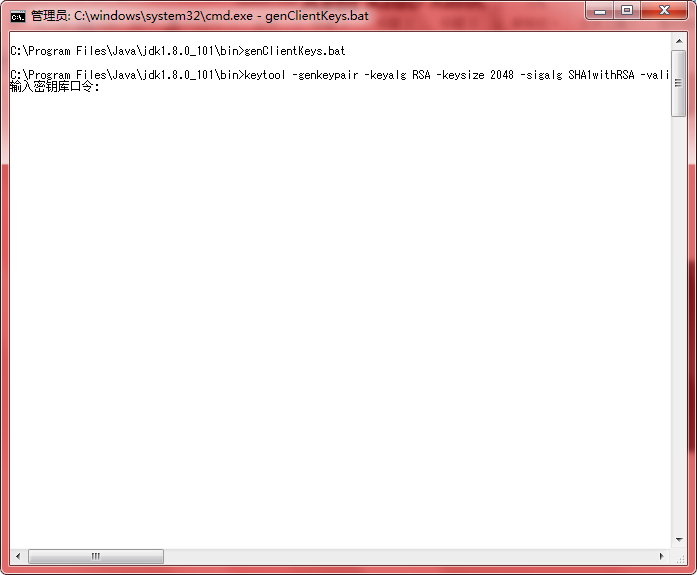
-keysize 2048

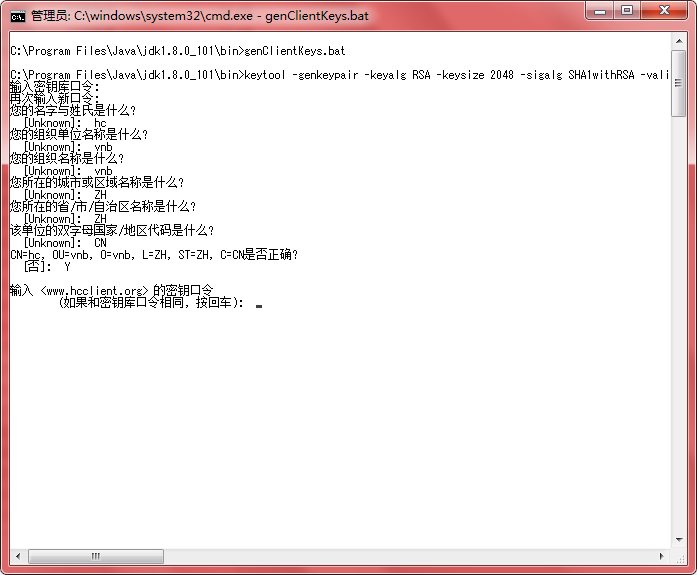
-sigalg SHA1withRSA

-validity 36000

-alias [www.hcclient.org](http://www.hcclient.org)

-keystore hcclient.keystore





20180917 client密钥口令qwerty



20180917 server 密钥口令 qwerty

### 使用OpenSSL

OpenSSL不仅可以对俄文件进行Base64编码、生成数字证书，同样可以进行消息摘要算法实现。