# Java实现数字签名

## 一 数字签名算法

　　带有秘钥(公钥,私钥)的消息摘要算法

　　验证数据完整性,认证数据来源,抗否认

　　OSI参考模型

　　私钥签名,公钥验证

　　RSA,DSA,ECDSA

## 二 数字签名算法RSA

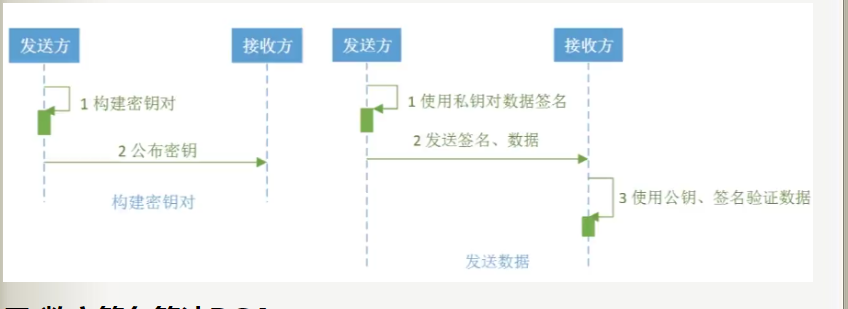
### 经典算法

　　MD, SHA两类

### 例子:jdkRSA 算法

|  |
| --- |
| package com.dzj.rsa;  import java.security.KeyFactory;  import java.security.KeyPair;  import java.security.KeyPairGenerator;  import java.security.PrivateKey;  import java.security.PublicKey;  import java.security.Signature;  import java.security.interfaces.RSAPrivateKey;  import java.security.interfaces.RSAPublicKey;  import java.security.spec.PKCS8EncodedKeySpec;  import java.security.spec.X509EncodedKeySpec;  import org.apache.commons.codec.binary.Hex;  public class RSA {  private static String src = "imooc security rsa";  public static void main(String[] args) {  jdkRSA();  }  public static void jdkRSA() {  try {  // 1 初始化密钥  KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance("RSA");  keyPairGenerator.initialize(512);  KeyPair keyPair = keyPairGenerator.generateKeyPair();  RSAPublicKey rsaPublicKey = (RSAPublicKey) keyPair.getPublic();  RSAPrivateKey rsaPrivateKey = (RSAPrivateKey) keyPair.getPrivate();  // 2 私钥加密、公钥解密——加密  // 用私钥进行签名  PKCS8EncodedKeySpec pkcs8EncodedKeySpec = new PKCS8EncodedKeySpec(rsaPrivateKey.getEncoded());  KeyFactory keyFactory = KeyFactory.getInstance("RSA");  // 执行key的转化  PrivateKey privateKey = keyFactory.generatePrivate(pkcs8EncodedKeySpec);  // 声明一个签名对象 使用JDK实现  Signature signature = Signature.getInstance("MD5withRSA");  signature.initSign(privateKey);  signature.update(src.getBytes());  byte[] sign = signature.sign();  System.out.println("私钥加密、公钥解密——加密:" + Hex.encodeHexString(sign));  // 3私钥加密、公钥解密——解密  X509EncodedKeySpec x509EncodedKeySpec = new X509EncodedKeySpec(rsaPublicKey.getEncoded());  keyFactory = KeyFactory.getInstance("RSA");  PublicKey publicKey = keyFactory.generatePublic(x509EncodedKeySpec);  // 创建签名对象  signature = Signature.getInstance("MD5withRSA");  signature.initVerify(publicKey);  signature.update(src.getBytes());  boolean verify = signature.verify(sign);  System.out.println("私钥加密、公钥解密——解密:" + verify);  } catch (Exception e) {  // TODO Auto-generated catch block  e.printStackTrace();  }  }  } |

## 数字签名算法----过程



## 三 数字签名算法DSA

　　DSS(Digital Signature Standard)数字签名标准

　　DSA(Digital Signature Algorithm)数字签名算法

　　DSA仅包含数字签名,没有办法进行加密通讯

　　RSA即包括加解密,又包括数字签名的算法

### 例子

|  |
| --- |
| package com.dzj.dsa;  import java.security.KeyFactory;  import java.security.KeyPair;  import java.security.KeyPairGenerator;  import java.security.PrivateKey;  import java.security.PublicKey;  import java.security.Signature;  import java.security.interfaces.DSAPrivateKey;  import java.security.interfaces.DSAPublicKey;  import java.security.spec.PKCS8EncodedKeySpec;  import java.security.spec.X509EncodedKeySpec;  import org.apache.commons.codec.binary.Hex;  public class DSA {  private static String src = "imooc security dsa";  public static void main(String[] args) {  jdkDSA();  }  public static void jdkDSA() {  try {  // 1.初始化密钥  KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance("DSA");  keyPairGenerator.initialize(512);  KeyPair keyPair = keyPairGenerator.generateKeyPair();  DSAPublicKey dsaPublicKey = (DSAPublicKey) keyPair.getPublic();  DSAPrivateKey dsaPrivateKey = (DSAPrivateKey) keyPair.getPrivate();  // 2.执行签名 私钥进行签名  PKCS8EncodedKeySpec pkcs8EncodedKeySpec = new PKCS8EncodedKeySpec(dsaPrivateKey.getEncoded());  KeyFactory keyFactory = KeyFactory.getInstance("DSA");  PrivateKey privateKey = keyFactory.generatePrivate(pkcs8EncodedKeySpec);  Signature signature = Signature.getInstance("SHA1withDSA");  signature.initSign(privateKey);  signature.update(src.getBytes());  byte[] result = signature.sign();  System.out.println("jdk dsa sign : " + Hex.encodeHexString(result));  // 3.验证签名  X509EncodedKeySpec x509EncodedKeySpec = new X509EncodedKeySpec(dsaPublicKey.getEncoded());  keyFactory = KeyFactory.getInstance("DSA");  PublicKey publicKey = keyFactory.generatePublic(x509EncodedKeySpec);  signature = Signature.getInstance("SHA1withDSA");  signature.initVerify(publicKey);  signature.update(src.getBytes());  boolean bool = signature.verify(result);  System.out.println("jdk dsa verify : " + bool);  } catch (Exception e) {  e.printStackTrace();  }  }  } |

## 四 数字签名算法ECDSA

　　微软

　　Ellipticc Curve Digital Signature Algorithm,椭圆曲线数字签名算法

　　速度快,强度高,签名短

### 例子:

|  |
| --- |
| package com.dzj.ecdsa;  import java.security.KeyFactory;  import java.security.KeyPair;  import java.security.KeyPairGenerator;  import java.security.PrivateKey;  import java.security.PublicKey;  import java.security.Signature;  import java.security.interfaces.ECPrivateKey;  import java.security.interfaces.ECPublicKey;  import java.security.spec.PKCS8EncodedKeySpec;  import java.security.spec.X509EncodedKeySpec;  import org.apache.commons.codec.binary.Hex;  public class ECDSA {  private static String src = "imooc security ecdsa";  public static void main(String[] args) {  jdkECDSA();  }  public static void jdkECDSA() {  try {  // 1 初始化密钥  KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance("EC");  keyPairGenerator.initialize(256);  KeyPair keyPair = keyPairGenerator.generateKeyPair();  ECPublicKey ecPublicKey = (ECPublicKey) keyPair.getPublic();  ECPrivateKey ecPrivateKey = (ECPrivateKey) keyPair.getPrivate();  // 2 执行签名  PKCS8EncodedKeySpec pkcs8EncodedKeySpec = new PKCS8EncodedKeySpec(ecPrivateKey.getEncoded());  KeyFactory keyFactory = KeyFactory.getInstance("EC");  PrivateKey privateKey = keyFactory.generatePrivate(pkcs8EncodedKeySpec);  Signature signature = Signature.getInstance("SHA1withECDSA");  signature.initSign(privateKey);  signature.update(src.getBytes());  byte[] sign = signature.sign();  System.out.println("jdk ecdsa sign:" + Hex.encodeHexString(sign));  // 3验证  X509EncodedKeySpec x509EncodedKeySpec = new X509EncodedKeySpec(ecPublicKey.getEncoded());  keyFactory = KeyFactory.getInstance("EC");  PublicKey publicKey = keyFactory.generatePublic(x509EncodedKeySpec);  signature = Signature.getInstance("SHA1withECDSA");  signature.initVerify(publicKey);  signature.update(src.getBytes());  boolean verify = signature.verify(sign);  System.out.println("jdk ecdsa verify:" + verify);  } catch (Exception e) {  // TODO Auto-generated catch block  e.printStackTrace();  }  }  } |