数字签名和数字证书

演讲者：谢攀

中科软科技股份有限公司
Sinosoft Co., Ltd.

演讲大纲

一、加密的基础知识

- 加密的基础概念
- 加密的分类

二、数字签名和数字证书

三、实际应用举例

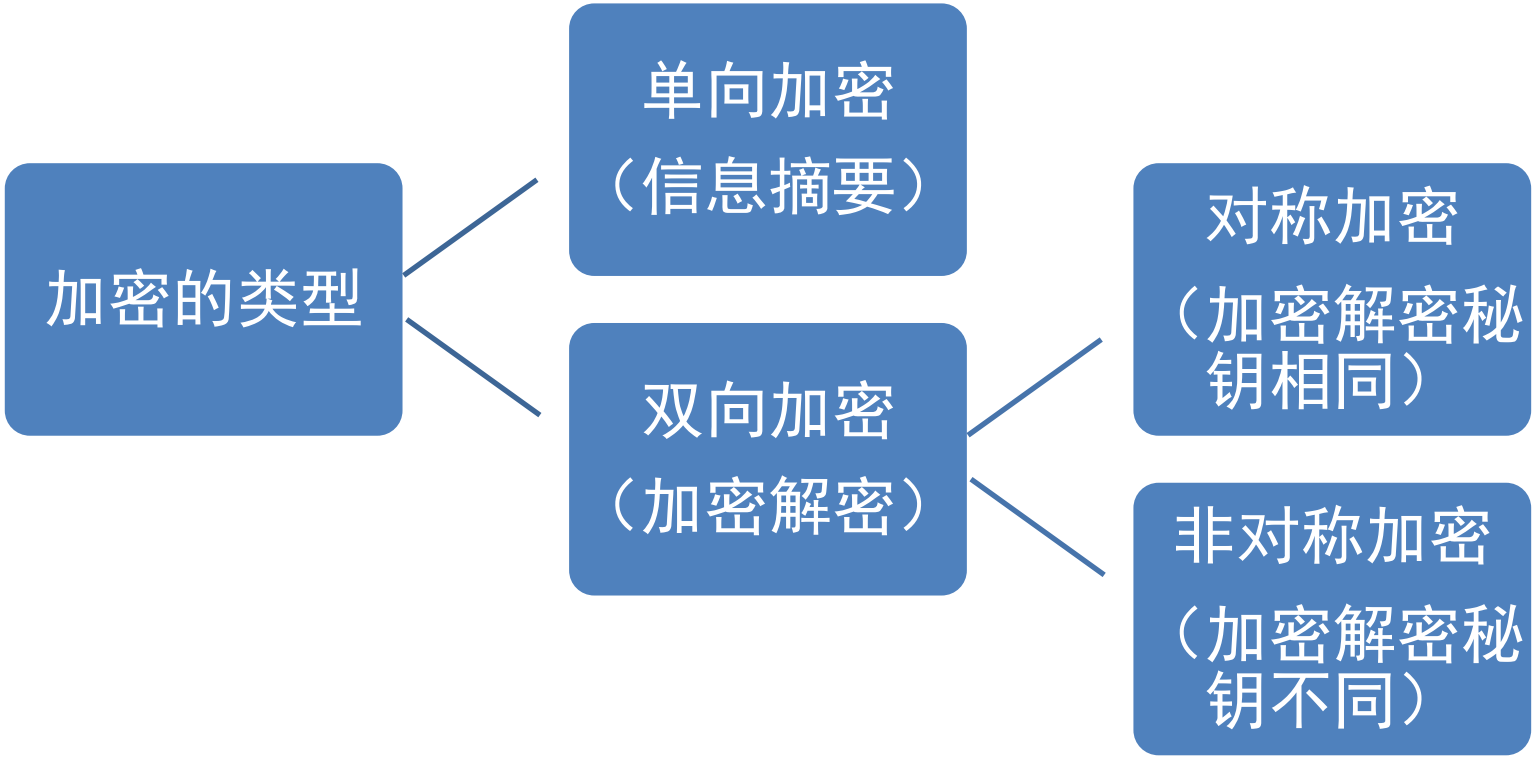
一、加密的基础知识

---加密的基础概念

- 明文-plaintext：被隐蔽的消息称作明文。
- 密文-ciphertext：隐蔽后的消息称作密文
- 加密-encryption：将明文变换成密文的过程称作加密
- 解密-decryption：由密文恢复出原明文的过程称作解密
- 加密算法：对明文进行加密时采用的算法
- 解密算法：对密文进行解密时采用算法
- 加密秘钥-encryption key和解密秘钥-decryption key：加密算法和解密算法的操作通常是在一组密钥key的控制下进行的。分别称为加密密钥和解密密钥。

一、加密的基础知识

---加密的分类



一、加密的基础知识

---单向加密（信息摘要）

- 通过算法将任意长度明文生成定长的密文，不可逆，又叫信息摘要（Message Digest）。主要作用是保证信息的完整性。
- 信息摘要是一个唯一对应一个消息或文本的**固定长度**的值，它由一个单向Hash加密函数对消息进行作用而产生。
- 信息摘要的主要算法包括MD2、MD5、SHA1、 SHA-256等。
- MD5：Message Digest Algorithm MD5（中文名为消息摘要算法第五版）为计算机安全领域广泛使用的一种散列函数，用以提供消息的完整性保护。

一、加密的基础知识

---单向加密（信息摘要）

主要用途

一致性验证

- 典型应用是对一段信息（Message）产生信息摘要（Message-Digest），以防止被篡改。
- 一般用来作程序或者文件的验证

数字证书

- 主要还是使用了信息摘要功能的防篡改的功能，配合第三方机构进行数字签名。

安全访问认证

- MD5还广泛用于操作系统的登陆认证上。
- 如在Unix系统中用户的密码是以MD5（或其它类似的算法）经Hash运算后存储在文件系统中。

一、加密的基础知识

---单向加密（信息摘要）

- Java实现（MD5算法为例）：
  - 涉及主要类：
    - **java.security.MessageDigest：摘要生成类**
  - 简单实现：
    - String a="abc";
      MessageDigest md=MessageDigest.getInstance("MD5");
      byte[] bi=a.getBytes();
      md.update(bi);
      System.out.println("字符:"+a+",摘要:“
      +new String(md.digest()));

一、加密的基础知识

---双向加密：对称加密

- 双向加密中通过明文加密后形成的密文，可以通过算法还原成明文，加密是可逆的。
- 采用单钥密码系统的加密方法，同一个密钥可以同时用作信息的加密和解密，这种加密方法称为对称加密，也称为单密钥加密。
- 算法是一组规则，规定如何进行加密和解密。因此对称式加密本身不是安全的。
- 常用的对称加密有：DES、IDEA、RC2、RC4、SKIPJACK、RC5、AES算法等。
- IDEA：International Data Encryption Algorithm

一、加密的基础知识

---双向加密：对称加密

主要用途

- 信息传送 • 消息传递加密
- 安全访问 • 用户密码认证

优点：算法公开、计算量小、加密速度快、加密效率高。
缺点：交易双方都使用同样钥匙，安全性得不到保证。无法进行身份认证。

一、加密的基础知识

---双向加密：对称加密

- Java实现（DES算法为例）：
  - 涉及主要类：
    - **javax.crypto.KeyGenerator：KEY生成器**
    - **javax.crypto.SecretKey：密钥**
    - **javax.crypto.Cipher：加密解密类**
  - 简单实现：
    - String b=“def”;
      KeyGenerator keyg=KeyGenerator.getInstance(“DES”);//KEY生成器
      SecretKey key=keyg.generateKey();//密钥
      Cipher c=Cipher.getInstance(“DES”);//加密解密类
      c.init(Cipher.ENCRYPT_MODE, key);//加密或者解密初始化
      String enstr=new String(c.doFinal(b.getBytes()));//加密
      System.out.println(“字符串：”+b+“ 加密了...:"+enstr);
      Cipher d=Cipher.getInstance("DES");
      d.init(Cipher.DECRYPT_MODE, key);//解密初始化
      System.out.println("解密了..."+
      new String(d.doFinal(c.doFinal(b.getBytes()))));

一、加密的基础知识

---双向加密：非对称加密

- 双向加密中通过明文加密后形成的密文，可以通过算法还原成明文，加密是可逆的。
- 与对称加密算法不同，非对称加密算法需要两个密钥：公开密钥（publickey）和私有密钥（privatekey）。
- 公开密钥与私有密钥是一对，如果用公开密钥对数据进行加密，只有用对应的私有密钥才能解密；如果用私有密钥对数据进行加密，那么只有用对应的公开密钥才能解密。因为加密和解密使用的是两个不同的密钥，所以这种算法叫作非对称加密算法。
- 常用算法：RSA、DSA…

一、加密的基础知识

---双向加密：非对称加密

主要用途

数字签名

- 自签名或第三方签名

数字证书

- 由CA机构，又称为证书授权（Certificate Authority）中心发行的，在网上用它来识别对方的身份

☆最大特点：进行身份认证

一、加密的基础知识

---双向加密：非对称加密

- Java实现（RSA算法为例）：
  - 涉及主要类：
    - **java.security.KeyPairGenerator：密钥对生成器**
    - **java.security.KeyPair：密钥对**
    - **java.security.interfaces.RSAPublicKey：公钥接口**
    - **java.security.interfaces.RSAPrivateKey：私钥接口**
    - **javax.crypto.Cipher：加密解密类**

一、加密的基础知识

---双向加密：非对称加密

- Java实现（RSA算法为例）：
  - 简单实现：
    - String me="abcd";
      KeyPairGenerator kpg=KeyPairGenerator.getInstance("RSA");
      KeyPair kp =kpg.generateKeyPair();
      RSAPublicKey rpubk=(RSAPublicKey)kp.getPublic();
      RSAPrivateKey rprik=(RSAPrivateKey)kp.getPrivate();
      Cipher c=Cipher.getInstance("RSA");
      c.init(Cipher.ENCRYPT_MODE,rpubk);
      byte[] enstr = c.doFinal(me.getBytes());
      String str =new String(enstr);
      System.out.println("明文是:" + me+“ 加密后是:" + str);
      Cipher d=Cipher.getInstance("RSA");
      d.init(Cipher.DECRYPT_MODE,rprik);
      System.out.println("解密后是:" + new String(d.doFinal(enstr)));

演讲大纲

一、加密的基础知识

二、数字签名和数字证书

- 数字签名
- 数字证书
- 数字证书的使用

三、实际中的应用举例

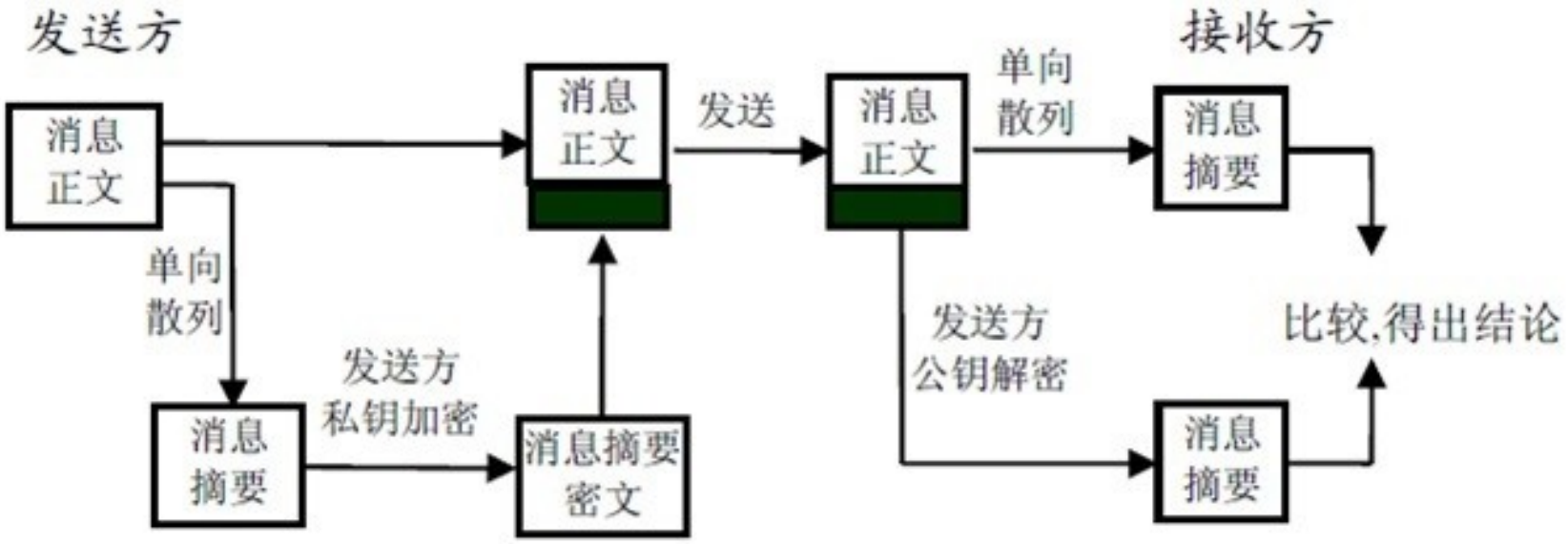
二、数字签名和数字证书

---数字签名

- 数字签名（又称公钥数字签名、电子签章）是一种使用了公钥加密领域的技术实现，用于鉴别数字信息的方法。一套数字签名通常定义两种互补的运算，一个用于签名，另一个用于验证。
- 数字签名是非对称密钥加密技术与数字摘要技术的应用。
- 进行数字签名至少应该进行以下2步：
  - （1）发方用单向散列函数F对消息正文M进行计算，产生散列码h
  - （2）发方用其用其私匙对散列码进行h加密，把加密后的散列码和消息正文一起发送出来。
- 验证数字签名至少需要以下2个步骤：
  - （1）接方用单向散列函数F对接受到消息正文M进行计算，产生散列码h2
  - （2）接方用发方的公钥对接收到散列码进行解密，还原得到散列码h,比较散列码h和散列码h是否一致。

二、数字签名和数字证书

---数字签名



数字签名的基本过程

二、数字签名和数字证书

---数字签名

- Java实现（用SHA1算法进行散列，用RSA算法进行加密）：
  - 涉及主要类：
    - **java.security.Signature：签名类**
  - 生成签名数据

```
Signature sign=Signature.getInstance("SHA1WithRSA");//实例化Signature
sign.initSign(privateKey);//设置加密散列码用的私钥，生成方法不再赘述
byte[] msg=new String("abc").getBytes();//新建一个消息，只能是byte数组
sign.update(msg);//设置散列算法的输入
byte[] data=null;//签名内容
data=sign.sign();//进行散列，对产生的散列码进行加密并返回
```

  - 验证信息

```
sign.initVerify(publicKey);//设置解密散列码用的公钥，生成方法不再赘述
sign.update(msg);//设置散列算法的输入
sign.verify(data);//进行散列计算，比较计算所得散列码是否和解密的散列
    码是否一致。一致则验证成功，否则失败
```

二、数字签名和数字证书

---数字证书

- 数字证书，又称为数码证书、数字证书、数字凭证、电子凭证或数字证书，是一种用于计算机的身分识别机制。数字证书不是数字身份证，而是身份认证机构盖在数字身份证上的一个章或印（或者说加在数字身份证上的一个签名），这一行为表示身份认证机构已认定这个持证人。
- 数字证书可向专门的CA机构申请，也可以生成没有认证的数字证书。
- 作为文件形式存在的证书一般有这几种格式：

1.带有私钥的证书由Public Key Cryptography Standards #12，PKCS#12标准定义，包含了公钥和私钥的二进制格式的证书形式，以pfx作为证书文件后缀名。

2.二进制编码的证书 证书中没有私钥，DER 编码二进制格式的证书文件，以cer作为证书文件后缀名。

3.Base64编码的证书证书中没有私钥，BASE64 编码格式的证书文件，也是以cer作为证书文件后缀名。

只有pfx格式的数字证书是包含有私钥的，cer格式的数字证书里面只有公钥没有私钥。

二、数字签名和数字证书

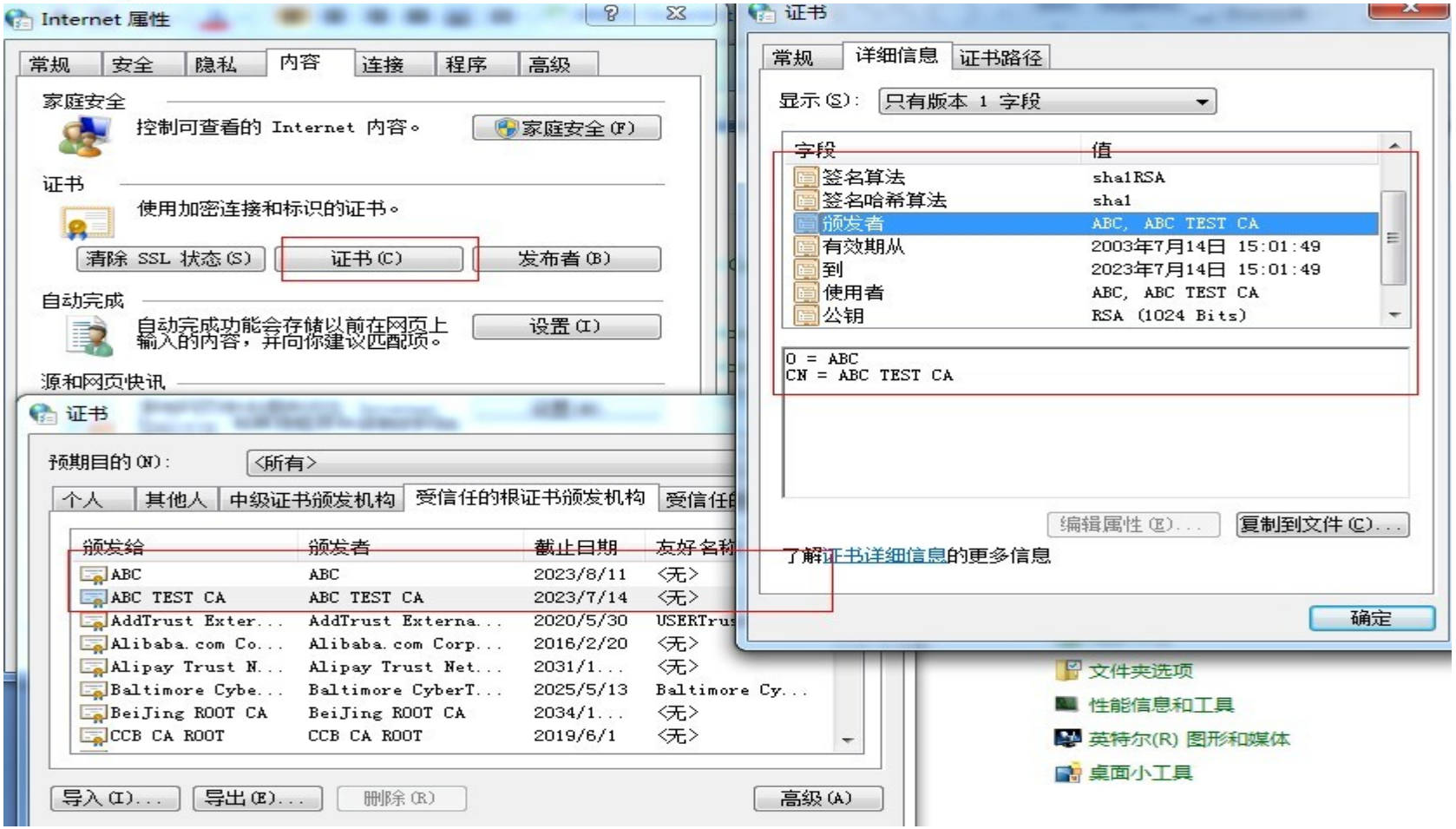
---数字证书

国际电信同盟 (ITU) X.509 标准规定的X.509 版本 3数字证书包含下列字段：

- **版本号** 证书所遵循的 X.509 标准的版本。
- **序列号** 唯一标识证书且由证书颁发机构颁发的编号。
- **证书算法标识** 证书颁发机构用来对数字证书进行签名的特定公钥算法的名称。
- **颁发者名称** 实际颁发该证书的证书颁发机构的标识。
- **有效期** 数字证书保持有效的时间段，并包含起始日期和过期日期。
- **使用者名称** 数字证书所有者的姓名。
- **使用者公钥信息** 与数字证书所有者关联的公钥以及与该公钥关联的特定公钥算法。
- **颁发者唯一标识符** 可以用来唯一标识数字证书颁发者的信息。
- **使用者唯一标识符** 可以用来唯一标识数字证书所有者的信息。
- **扩充信息** 与证书的使用和处理有关的其他信息。
- **证书颁发机构的数字签名** 使用证书算法标识符字段中指定的算法以及证书颁发机构的私钥进行的实际数字签名。

二、数字签名和数字证书

---数字证书



二、数字签名和数字证书

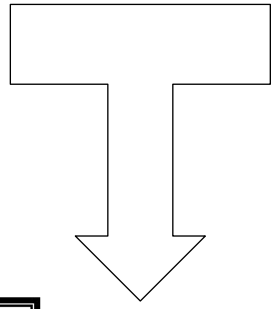
---数字证书

颁发过程：

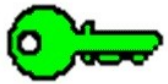
CA认证机构

CA的公钥

CA的私钥



鲍勃



鲍勃的公钥

鲍勃的私钥

CA 的私钥

数字

证书

二、数字签名和数字证书

---数字证书

- Java实现：
  - 涉及主要类：
    - **java.security.KeyStore**：此类表示密钥和证书的存储设施
  - Java实现（通过KeyStore存储证书）
    - KeyTool工具可生成,导入,导出数字证书,支持的证书的格式为当前流行的
    - 生成证书语法：keytool -genkey -validity 36000 -alias mykey
      -keyalg RSA –keystore d:\mykeystore
      - 执行这个命令后，后台会提示输入证书相关的信息。
      - 输入完成后，会生成d:\ mykeystore的文件，文件中存有自签名的证书，可直接导出。
    - 导出证书：keytool -export -alias mykey -file MJ.cer –keystore mykeystore
    - 导入证书：keytool -import -alias abc -file ABCCA.cer -keystore mykeystore

二、数字签名和数字证书

---数字证书

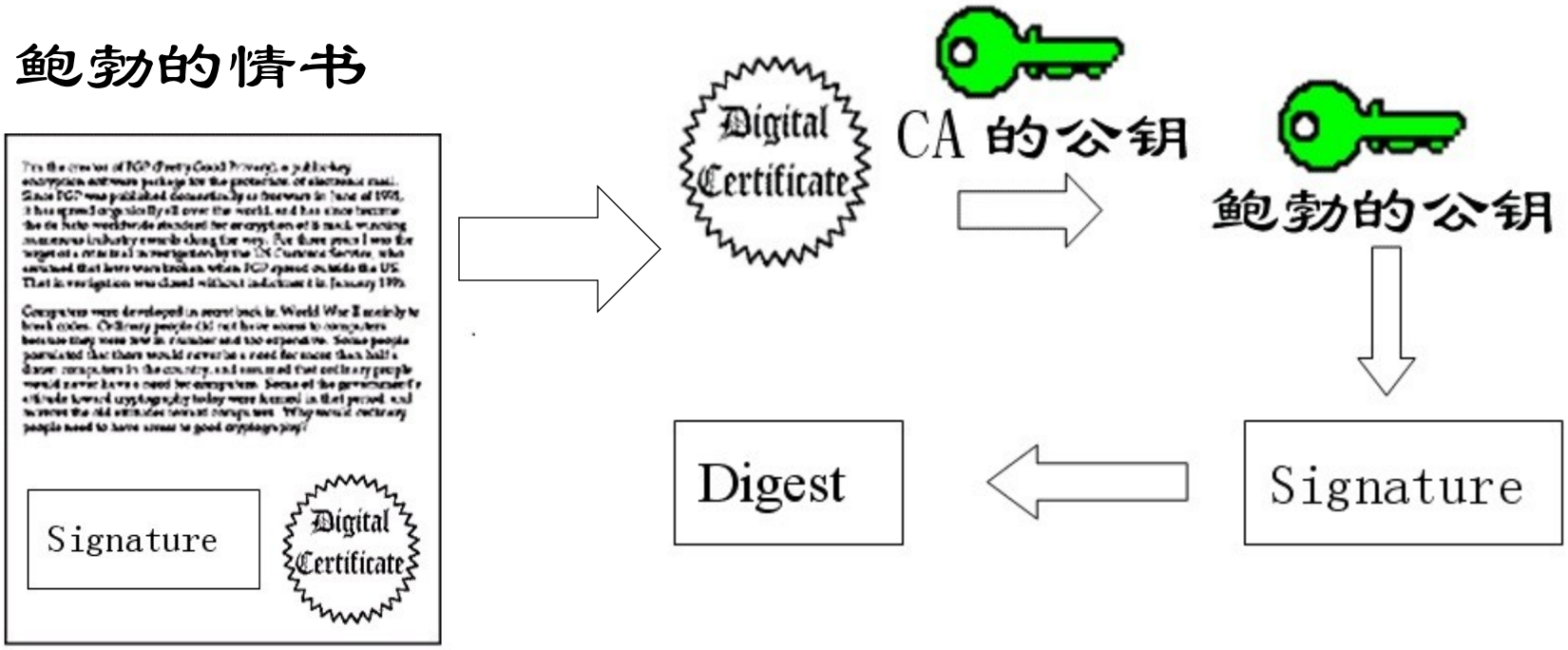
执行这个命令后，后台会提示输入证书相关的信息。

```
输入keystore密码:
再次输入新密码:
您的名字与姓氏是什么?
  [Unknown]:  www.zlex.org
您的组织单位名称是什么?
  [Unknown]:  zlex
您的组织名称是什么?
  [Unknown]:  zlex
您所在的城市或区域名称是什么?
  [Unknown]:  BJ
您所在的州或省份名称是什么?
  [Unknown]:  BJ
该单位的两字母国家代码是什么
  [Unknown]:  CN
CN=www.zlex.org, OU=zlex, O=zlex, L=BJ, ST=BJ, C=CN 正确吗?
  [否]:  Y

输入<tomcat>的主密码
        (如果和 keystore 密码相同，按回车):
再次输入新密码:
```

二、数字签名和数字证书

---数字证书的使用



二、数字签名和数字证书

---数字证书的使用

- Java实现：
  - 涉及主要类（部分之前显示过的类省略）：
    - **java.security.cert.X509Certificate**：X.509 证书的抽象类。此类提供了一种访问 X.509 证书所有属性的标准方式。
    - **java.security.cert.CertificateFactory**：此类定义了用于从相关的编码中生成证书、证书路径 (CertPath) 和证书撤消列表 (CRL) 对象的 CertificateFactory 功能。
  - Java简易实现（此处略过，只说几个主要的方法和构造函数）
    - X509Certificate cert = (X509Certificate)keyStore.getCertificate(“mykey”);
      - //从现有keystore中获取别名为mykey的证书
    - cert.getSignature() ;//获取证书的原始签名
    - cert.getPublicKey() ;//获取证书的公钥

演讲大纲

一、加密的基础知识

二、数字签名和数字证书

三、实际中的应用举例

- Jar文件的数字签名
- https协议

三、实际中的应用举例

---Jar文件的数字签名

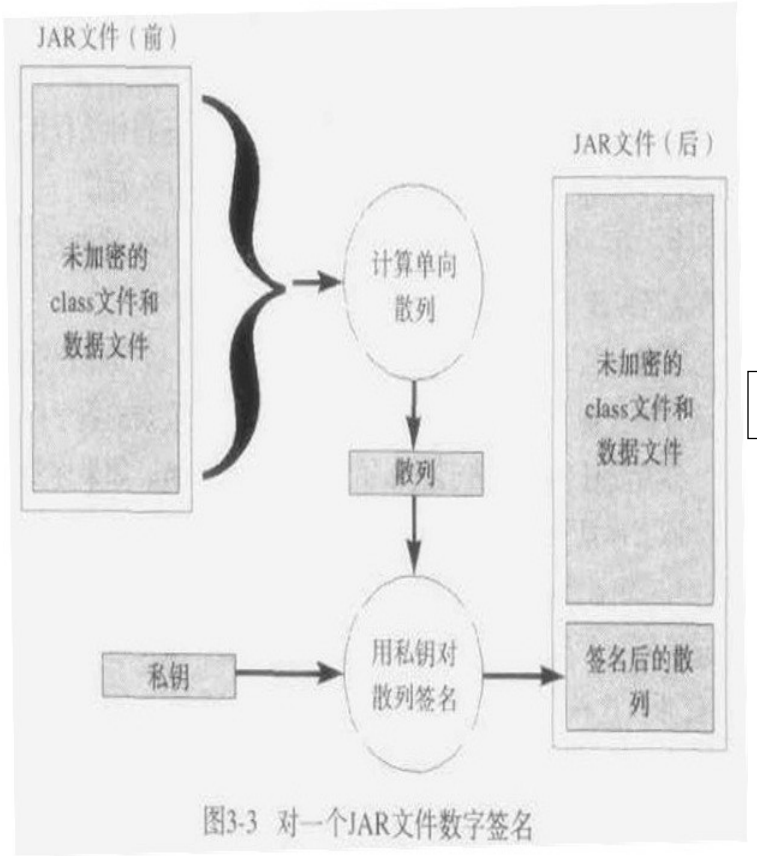


图3-3 对一个JAR文件数字签名

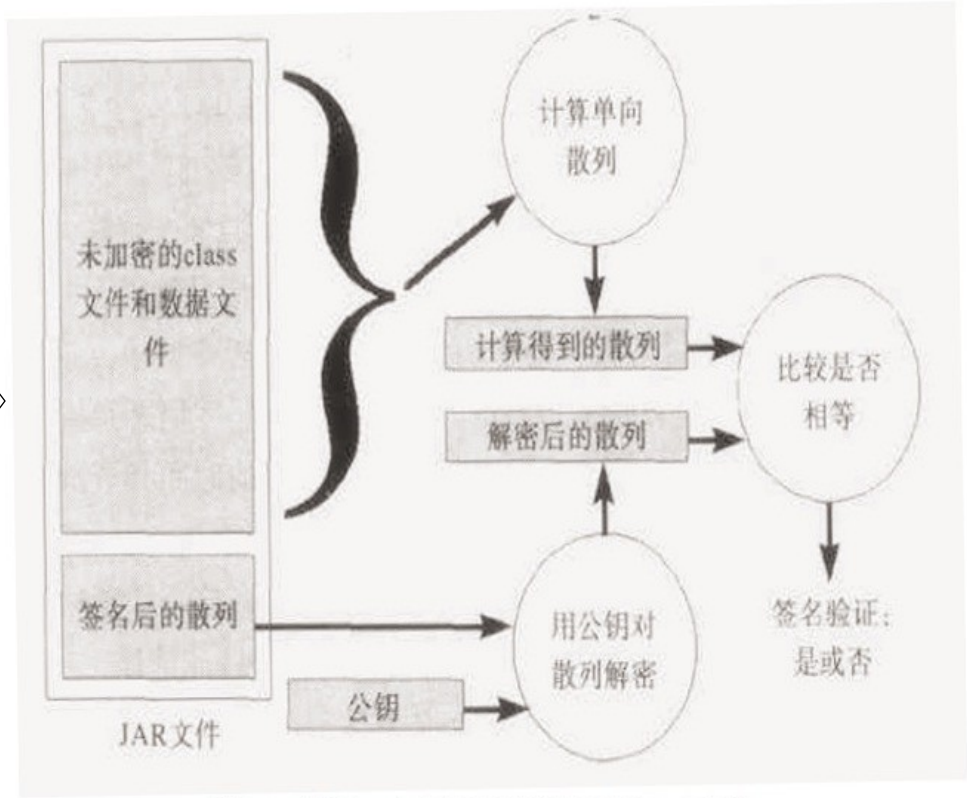


图3-4 认证一个经过数字签名后的JAR文件

二、数字签名和数字证书

---数字证书的使用

- Java实现：
  - 主要工具jarsigner：
    - 为 Java 归档 (JAR) 文件签名
    - 校验已签名的 JAR 文件的签名和完整性
  - 为JAR文件签名
    - jarsigner -keystore [keystoreFile] [jarfile] [alias]
    - 例如：jarsigner -keystore mykeystore test.jar mykey
  - 验证一个jar文件
    - jarsigner -verify jarfile
    - 例如：jarsigner -verify test.jar

三、实际中的应用举例

---https协议

客户端向服务器发出加密请求

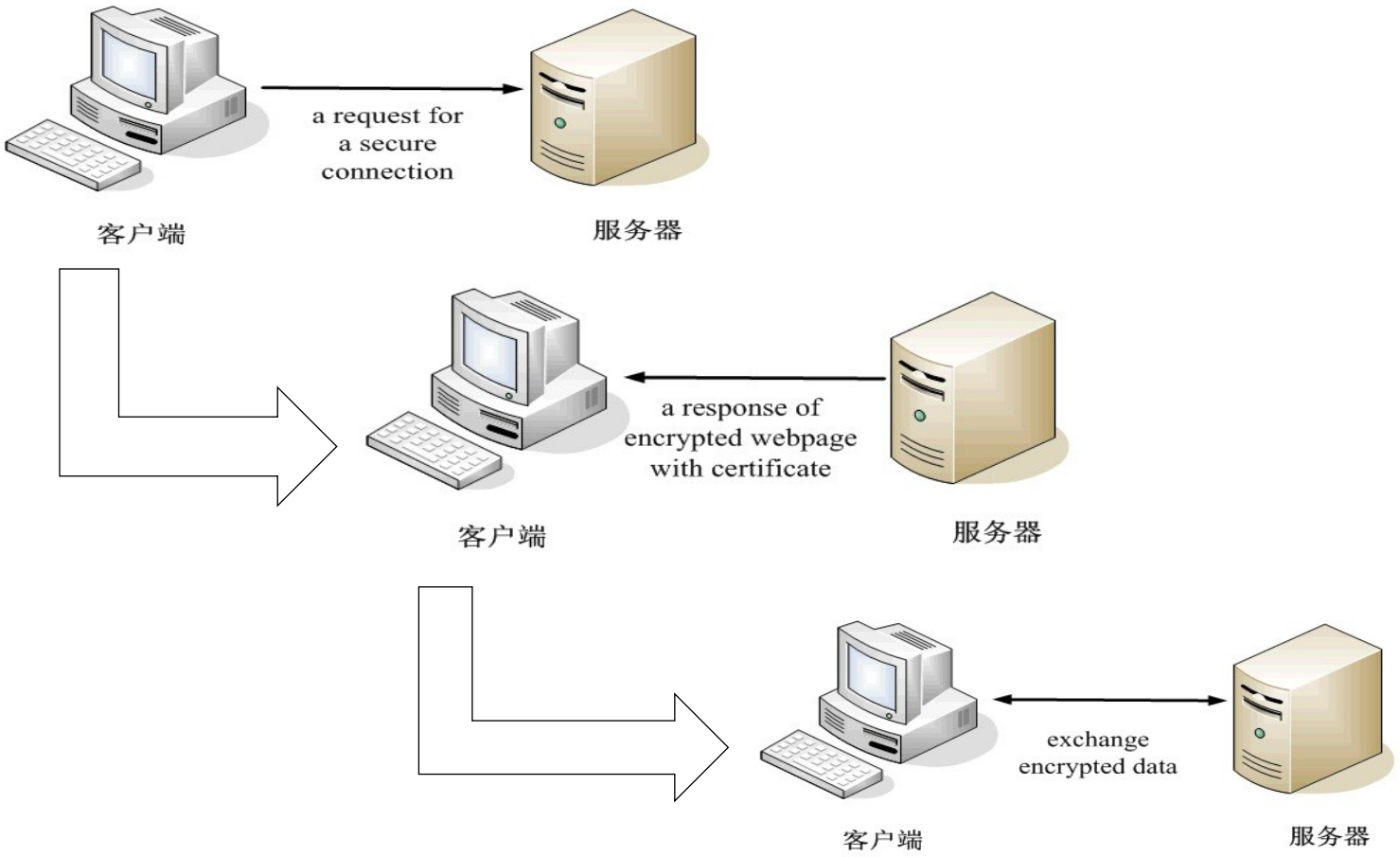
服务器用自己的私钥加密网页以后，连同本身的数字证书，一起发送给客户端。

客户端（浏览器）的"证书管理器"，有"受信任的根证书颁发机构"列表。客户端会根据这张列表，查看解开数字证书的公钥是否在列表之内。

如果数字证书是可靠的，客户端就可以使用证书中的服务器公钥，对信息进行加密，然后与服务器交换加密信息。

三、实际中的应用举例

---https协议



三、实际中的应用举例

---https协议

客户端向服务器发出加密请求

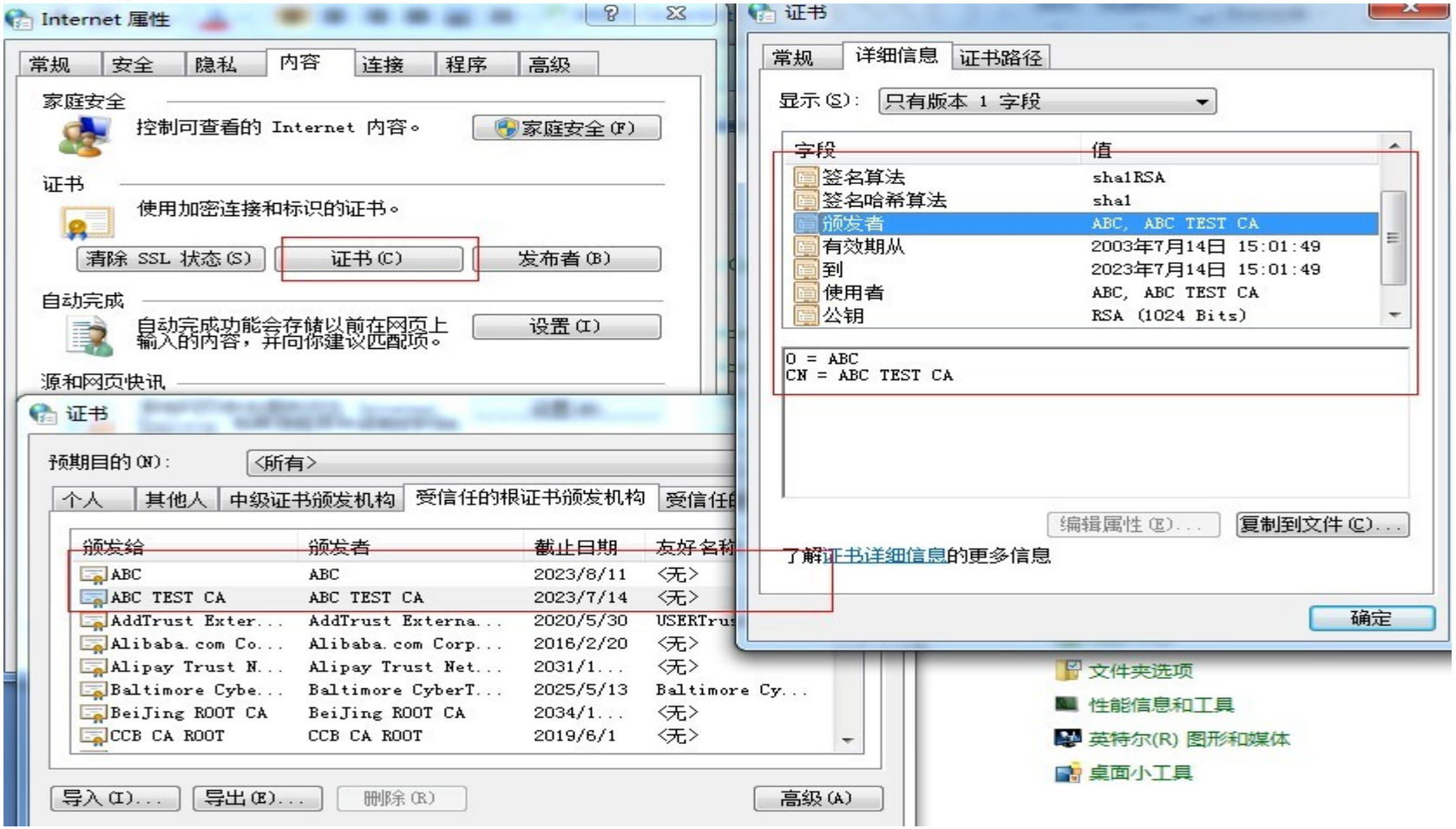
服务器用自己的私钥加密网页以后，连同本身的数字证书，一起发送给客户端。

客户端（浏览器）的"证书管理器"，有"受信任的根证书颁发机构"列表。客户端会根据这张列表，查看解开数字证书的公钥是否在列表之内。

如果数字证书是可靠的，客户端就可以使用证书中的服务器公钥，对信息进行加密，然后与服务器交换加密信息。

三、实际中的应用举例

---https协议



三、实际中的应用举例

---https协议

三、实际中的应用举例

---https协议

- **相关资料**
  - **数字证书简介（这个目录下的所有博文）**
  - **http://hubingforever.blog.163.com/blog/static/171040579201189812197 05/**
  - **数字签名是什么？**
  - http://mp.weixin.qq.com/s?__biz=MjM5NzA1MTcyMA==&mid=200055495&idx=1&sn=17af5e0699068f2ea84451c4865e5b43#rd
  - JAVA中常用的加密方法
  - http://www.iteye.com/topic/1122076


我有疑问

Thank You!