

文章编号:1001-9081(2005)02-0459-04

移动电子拍卖系统加密和签名特殊技术

乔少杰¹, 彭 舰¹, 郑学强², 林红君³

(1. 四川大学 计算机学院, 四川 成都 610064; 2. 解放军信息工程大学 信息工程学院, 重庆 450002;

3. 中电科技集团 第三十研究所, 四川 成都 610041)

(qiaoshaojie@cs.scu.edu.cn)

摘 要:在实现安全的移动电子拍卖系统的实践经验基础上,分析了系统的安全性需求,介绍了移动电子拍卖系统加密和签名特殊技术,包括密钥的生成、存储和更新,服务器端对 SOAP 消息的加密、解密以及自签名证书密钥模型,客户端加密引擎和签名引擎的实现。该系统可以成功地移植到移动设备中,提高了无线 Web 服务的安全性。

关键词:移动电子拍卖系统;加密;签名;SOAP

中图分类号: TP309.7 **文献标识码:** A

Special encryption and sign techniques in mobile auction system

QIAO Shao-jie¹, PENG Jian¹, ZHENG Xue-qiang², LIN Hong-jun³

(1. School of Computer Science and Engineering, Sichuan University, Chengdu Sichuan 610064, China;

2. School of Information Engineering, PLA Information Engineering University, Chongqing 450002, China;

3. No. 30 Institute, China Electronics Technology Group Corporation, Chengdu Sichuan 638550, China)

Abstract: Based on the implements of the secure mobile auction system, the paper analysed the security requirements of this system, introduced the special encryption and sign techniques in mobile auction system, these techniques contained the making, storing and updating of the secret key, the encryption, decryption and the self-sign certificate model of SOAP message in server, the implement of the encryption engine and the sign engine in client. This system can be successfully transplanted to mobile devices in order to improve the security of the wireless Web Services.

Key words: mobile auction system; encryption; sign; SOAP

0 引言

因特网与移动通信技术的高度融合促进了移动电子商务的产生。利用现有的因特网技术和移动通信技术,在这两大平台上开发移动商务应用成为现实。尽管移动商务为人们带来了全新的便捷的服务方式,但是目前现有的移动商务应用广泛存在安全漏洞,使得应用的普及受到极大的限制。

在技术方面,基于 Java 的 Web 服务和无线 Java 开发技术的逐步成熟,为实现移动电子应用提供了技术上的支持。但当前技术仍不成熟。安全性仍是待解决的问题之一。无线通信是无线电波拦截容易获取的目标,而无线设备几乎没有任何计算能力来支持所有通信数据的强加密。此外,在后端,Web 服务运行在企业防火墙之外并使用开放消息传递协议来彼此交互。无线 Web 服务同样是易遭受各种破解攻击的目标。

因此研究基于 J2ME 平台技术的安全性和 Web 服务技术的移动电子商务的安全性具有很大的现实意义。

1 系统简介及其安全性需求分析

下面将针对我们开发的基于 J2ME 的移动拍卖系统——

“拍趣”拍卖系统(该系统已经成功的移植到手机中)具体地分析一个无线 Web 服务平台的安全技术实现的案例。

1.1 系统简介

本系统前台采用 J2ME 平台实现应用,后台采用 Web 服务技术实现服务提供者。

移动电子拍卖系统由以下几个部分组成:

- 1) 系统初始化:后台 Web 服务启动,并公布有关拍卖品的信息;
- 2) 用户注册:每个竞拍者发送其必要信息到服务器端进行注册;
- 3) 用户身份:用户在系统中可以拥有双重身份,既可以作为卖家,也可以作为买家;
- 4) 用户获取拍卖信息:拍卖管理程序提供等待拍卖的商品信息,系统用户可以登录后浏览这些信息;
- 5) 拍卖结束:拍卖成功,买家得到卖家的联系方式,系统通知卖家竞拍过程结束;
- 6) 拍卖失败:通知卖家,竞拍结束。

1.2 系统安全性需求分析

在电子拍卖过程中要保证竞拍者注册信息的保密性及保护竞拍者的出价信息,任何人都无法获得竞拍者的身份及其

收稿日期:2004-07-23;修订日期:2004-10-11 基金项目:四川省青年软件创新工程(2003AA0003)

作者简介:乔少杰(1981-),男,黑龙江牡丹江人,硕士研究生,主要研究方向:数据库与知识工程、数据挖掘; 彭舰(1970-),男,四川成都人,副教授,博士,主要研究方向:分布式处理与中间件技术; 郑学强(1974-),男,重庆人,硕士研究生,主要研究方向:网络与信息技术; 林红君(1982-),女,四川岳池人,主要研究方向:计算机应用技术。

投标信息。

电子拍卖系统应有的安全特性如下:

竞拍者的信息保密性: 系统提供对注册用户的信息以及出价信息的安全传输,防止被第三方截获。

竞价信息的数据完整性: 用户在出价的时候,客户端应用实现对用户出价信息的签名操作,服务器端可以通过验证签名信息保证数据的完整性。

竞拍服务的真实性: 系统为客户端提供了验证签名的应用,对服务器端的数据进行验证,保证接受的服务的真实性。

不可欺骗性: 任何人都不能伪装某个已注册的竞拍者进行竞价。

2 系统安全架构

本系统采用单向认证方案,系统的安全架构如图 1 所示。

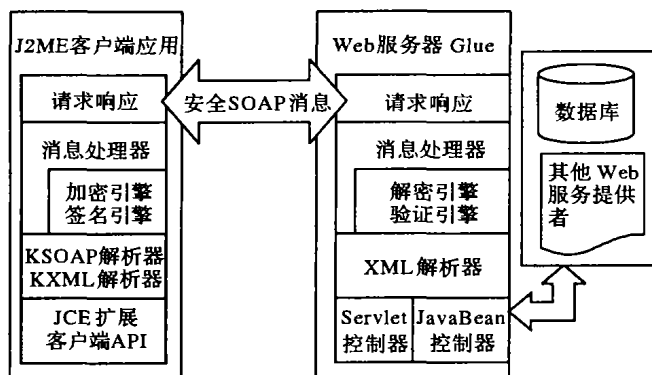


图1 无线电子拍卖应用安全架构

系统包括使用对称算法实现加密和非对称算法签名两个阶段过程。

系统在服务器端将产生一个共享密钥,用于使用 Triple DES 对称算法^[1]来加密/解密数据。对称算法非常有效,并且可以与用于加密和解密计算的单一密钥一起使用。Web 服务安全性 (WS-Security) 实现使用一个随机生成的密钥。一旦消息的数据被加密,密钥本身就被插入到消息中。

系统在客户端应用中保存了系统用于加密/解密 SOAP^[2]消息的共享密钥。在对 SOAP 消息操作的时候使用。对于消息的签名使用的是一对非对称密钥——一个私有的和一个公共的。这里使用 X.509 证书格式的密钥,证书中包含有两个密钥,一个是证书的所有者私有的,而另一个是与他们一起开展业务的其他人共享的。

客户端应用对 SOAP 消息利用私有密钥进行数字签名。这里对 SOAP 消息体进行签名。经过客户端应用处理的签名消息,包括关于签名的密钥信息就作为二进制安全性令牌包含进 SOAP 头了。

当服务提供者接收到 Web 服务请求时,基于请求者的 URL,请求就定向到 SOAP 处理引擎(SOAP 运行时)。在请求中传送的消息数据是经过签名和加密的,所以第一步就是识别 SOAP 头中引用的 X.509 证书^[3,4],并从 X.509 证书链的 keystore(相当一个数据库)中检索它的私钥。一旦获得公钥,就可以使用非对称算法来验证签名。通过共享密钥对称算法来解密消息数据。

3 密钥的产生、存储和更新

服务器端产生密钥后存放在应用中,这里假设服务器端是可信的。服务器端仅提供了对来自客户端的数据的完整性验证。

首先在服务器端产生密钥,提取密钥参数,保留参数值,在发布客户端应用的时候,提取密钥参数值,重新构建密钥,针对所有的客户端,都使用同一对密钥。这一对密钥在服务器端的密钥库中有同样的备份,服务器端收到来自客户端的签名 SOAP 消息后,利用公钥验证消息的完整性。

提供加密的共享密钥也以密文的形式存储在客户端应用中,在对原始消息加密时使用。

现在需要导出密钥参数。系统首先在服务器端应用中产生用于签名的密钥对和用于加密的共享密钥对后,需要提取出密钥参数,利用这些参数,在客户端能够重新构建密钥对。获取密钥对中私钥的一个参数的方法如下:

```
privKey = (RSAPrivateCrtKeyParameters)
keyPair.getPrivate();
//导出私钥参数
BigInteger privateKeyDP = privKey.getDP();
//在控制台输出密钥参数
System.out.println("privateKeyDP = " + privateKeyDP);
//将控制台输出的密钥参数重新赋值构造新的密钥
privateKeyDP = new BigInteger
("2979750878679664390023599054709331027032702302444708
816092225306357352950956680073014685040182357508935392
476975479182826100362278654349880244458787077073");
```

私钥共有八个参数,需要一一导出并存储,保留在程序中,在客户端重新构建密钥。

如果密钥被长时间存储在服务器端而得不到更新,势必会造成密钥泄密。因此为了防止这种危险情况的出现,密钥应该定时更换。其中有一个需要注意的问题就是密钥的更新时间,无论是签发者或是被签者的密钥作废时间,要与每个证书的有效截止日期保持一致。

此系统采用了如下的技术解决密钥更新的问题,即 PKI (Public Key Infrastructure) 实体在密钥截止之前,就取得新密钥对和新证书,在截止日期到达后,PKI 中的实体便开始使用新的私钥进行对数据的签名,同时将旧密钥对和证书归档保存。

4 服务器端安全实现

4.1 安全服务内容

1) 用户注册服务

加密传输: 客户端提交注册信息后,消息处理器从原始消息中提取出消息体,针对消息体进行加密,加密的 SOAP 消息重新封装成一个完整的加密消息发送。

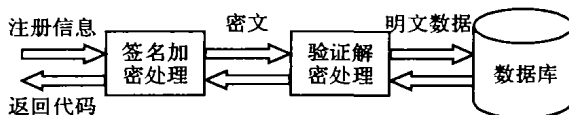


图2 用户注册服务

解密数据: 在服务器端对用户注册信息进行解密后提交给下一层逻辑。

签名注册信息: 对用户注册信息进行签名,到服务器端验

证数据的完整性。

用户注册服务的过程如图2所示。

2) 竞拍服务

签名出价数据:对用户提供的出价数据,使用客户端的私钥进行签名后发送,数据发送到服务器端后验证客户端消息的完整性。

加密/解密出价数据:客户端对出价数据加密,服务器端解密出价数据。

竞拍服务的过程如图3所示。

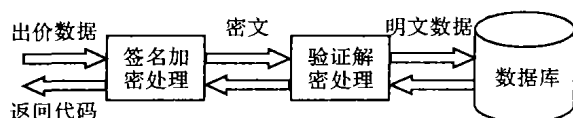


图3 竞拍服务

4.2 自签名证书密钥模型

在服务器端,系统采用 X.509 格式的证书的方式来存储密钥,在 JDK1.4 中,提供了一个命令行的工具 Keytool,它是安全密钥与证书的管理工具,管理一个存储了私有密钥和验证相应公共密钥以及它们相关联的 keystore。Keytool 把密钥和证书存储到一个 keystore 中。默任的实现 keystore 的是一个文件。它用一个密码保护密钥。

Keystore 有两个不同的入口:

1) 钥匙入口:保存了非常敏感的加密的钥匙信息,并且是用一个保护的格式存储以防止未被授权的访问。以这种形式存储的钥匙是秘密钥匙,或是一个对应证书链中公有钥匙的私有钥匙。

2) 信任证书入口:包含一个属于其他部分的单一公共钥匙证书。它之所以被称为“信任证书”,是因为 keystore 信任的证书中的公共钥匙真正属于证书所有者的身份识别。

使用 JDK1.4 自带的工具 KeyTool 可以生成一个自签名的符合 X.509 格式的证书。

生成证书的方法如下:

```
--> keytool -genkey -alias test -keyalg RSA -keystore
servercert.jks (指定 keystore 文件名). 输入 keystore 密码:
keystoretest
```

您的名字与姓氏是什么?

[Unknown]: Mobile Auction

您的组织单位名称是什么?

[Unknown]: College of Computer

您的组织名称是什么?

[Unknown]: Sichuan University

您所在的城市或区域名称是什么?

[Unknown]: Chengdu

您所在的州或省份名称是什么?

[Unknown]: Sichuan

该单位的两字母国家代码是什么?

[Unknown]: CN

CN = Mobile Auction, OU = College of Computer,

O = Sichuan University, L = Chengdu, ST = Sichuan,

C = CN 正确吗?

[否]: y

输入 <mykey> 的主密码

(如果和 keystore 密码相同,按回车):

4.3 验证服务实现

在竞拍服务中,系统提供了对来自客户端的已签名的出

价数据的验证,以保证数据的完整性。用户同时提供用户名和密码,系统验证用户的身份,证明用户出价数据的不可否认性。

服务器端进行验证签名服务的部署方法如下(在其部署描述符文件 Bid.xml 中加入下面显示为斜体的代码):

```
<service>
  <constructor>
    <class> cn.edu.cdutscu.ws.Bid </class>
  </constructor>
  <publish> yes </publish>
  <!-- 验证来自客户的针对消息体进行签名的消息 -->
  <wssSignature>
    <reference> soap:Envelope/soap:Body </reference>
    <trustStore> servercert.jks </trustStore>
    <trustStorePassword> keystoretest </trustStorePassword>
  </wssSignature>
</service>
```

4.4 基于 Glue 的 Triple DES 解密服务实现

在系统中已经开发的用户注册服务 User 中,需要实现对来自客户端的用户注册信息的解密功能。系统使用的 Web 服务器是 Glue5.01, Glue5.01 企业版已经实现了 WS-Security1.0 规范,支持 XML 的签名和加密,它内置的安全包提供了实现方案。在服务器端对 User 服务进行安全配置的方法如下:

首先需要为加密/解密应用产生一个共享密钥,在应用中可以通过实现产生共享密钥,这里系统使用 Glue 提供的一个可以产生基于 Triple DES 算法的对称密钥的工具来产生一个随机的用于加密/解密的对称密钥:

```
--> java TripleDESKey
kn89W7byFYz4EOXWs0UxnpJ/PVu28hWM
```

客户端利用此密钥加密 SOAP 消息,服务器端利用它解密 SOAP 消息。

针对用户注册服务,在其配置描述符文件 User.xml 中配置解密服务的方法如下:

```
<service>
  <constructor>
    <class> cn.edu.cdutscu.ws.User </class>
  </constructor>
  <publish> yes </publish>
  <!-- 来自客户端请求的 SOAP Body 的内容是使用此密钥加密的 -->
  <wssEncryption>
    <reference> soap:Envelope/soap:Body/* </reference>
    <secretKey>
      <realm> acl </realm>
    </secretKey>
  </wssEncryption>
</service>
```

在标签 <realm> 中表明用户所处的域。在 Glue5.01 中,使用了域的概念来实现对基于用户名/密码信息/角色信息的管理控制,在 Glue 中定义了基本域 (BasicRealm)、Acl 域 (AclRealm) 和 JAAS 域三种域,分别定义了不同级别的用户/角色类型,来控制客户端用户对资源的访问,这里采用了 Acl 域模式来实现访问控制。系统启动的时候会初始化域信息,域信息保存在一个特定的文件中,Acl 域的成员信息保存在

acl.xml 文件中,这个文件保存在 < Glue Home > \youwebapp\WEB-INF/security/目录下。

这里 Glue 使用访问控制结合共享密钥加密的机制来实现加密解密方案。在客户端必须是一个存在于 Acl 域的成员来访问服务器端的资源,在域成员中定义了共享密钥的值。<reference> 标签中的内容控制对消息体进行加密,标明为 soap:Envelope/soap:Body/* 则表示对 SOAP Body 进行解密,来自客户端的加密请求通过消息处理器解析后,在服务器端利用 Triple DES 算法生成的对称密钥对密文进行解密操作。

5 客户端安全实现

5.1 实现方案

要在无线设备上面实现签名和加密方案,在技术实现上将会遇到很多困难,和传统的有线网络上的方案相比有很大的不同。在本系统客户端要实现加密和签名应用,需要综合考虑下面几个因素:密钥的存储,SOAP 消息的提取和解析,SOAP 消息的封装和发送,算法的选择。

5.2 J2ME 客户端和 Web 服务交互

系统采用 KSOAP^[5] 的支持来处理底层的 SOAP 消息,在 J2ME Web Service 规范出现之前。要在 J2ME 客户端使用 Web 服务,一般通过两种方式:一是用 Servlet 作代理,二是用 KSOAP 在 MIDlet 中直接访问 Web 服务。本系统应用 KSOAP 对 SOAP 消息进行解析和封装。在无线设备上利用 KSOAP 调用 Web 服务的方法如下:

```
// 创建一个新的 SOAP 请求消息
SoapObject method = new SoapObject
    ("yourNamespace", "newUser");
// 添加方法的参数
method.addProperty("newUserParam",
    "userName: password: 12345");
// 创建一个连接对象
transport.HttpTransport transport =
    new ksoap.transport.HttpTransport("http://localhost: 8004/glue/
        services/User", "");
// 调用的到返回结果
Object result = transport.Call(transport, method);
```

5.3 加密引擎的实现

客户端对未发送的 SOAP 消息进行加密,重新构造一个加密的 SOAP 消息。首先从 SOAP 请求中得到将要发送的请求 SOAP 消息,获取 SOAP 消息的方法如下:

```
// 创建一个 SOAP 消息对象
SoapObject method = new SoapObject
    ("yourNamespace", "newUser");
method.addProperty("newUserParam", "hhh: hhh: 12345");
// 利用重写的 HttpTransport 类建立连接
ksoap.transport.HttpTransport transport =
    new ksoap.transport.HttpTransport
    ("http://star: 8004/glue/services/User", "");
```

系统提供了 KsoapUtils 类,在其中提供了一个 secureCall 方法,利用这个方法替代正常的 Call 调用,使得发送的 SOAP 消息在服务器端能够被验证和解密。

在应用通过 HttpTransport 连接调用远程服务之前,需要产生 SOAP 请求消息并加密 SOAP Body。

5.4 签名引擎的实现

对原始 SOAP 消息实现签名,首先要处理 SOAP 消息的摘要。产生消息摘要的方法如下,这里使用的产生消息摘要的算法是 SHA1 算法^[3,6]。

用于签名的私钥在应用中已经被重新构建,所以可以直接使用其对消息签名。签名的方法如下:

```
static public String getSignature (String bodyXml)
{
    SHA1Digest digestFac = new SHA1Digest();
    RSASigUtil ru = new RSASigUtil();
    digestFac = (SHA1Digest) ru.getDigest( bodyXml);
    RSAEngine rsaFac = new RSAEngine();
    PSSSigner signer = new PSSSigner
        (rsaFac, digestFac, 64);
    signer.init( true, privKey);
    byte [] signed = signer.generateSignature();
    String result = new String( Base64.encode(signed));
    return result;
}
```

6 结语

本文将安全模型实施到无线网络中,针对基于 HTTP 层之上的 SOAP 消息传输层,在服务器端应用 WS-Security 规范实现了传输层安全。本系统经过实验,增强了自身的安全性,对应用的实施和普及提供了安全保障。

但本系统仍然存在不足。速度是影响此应用的一个重要的因素,如果能够改进系统,只对部分信息进行加密而不是对整个消息体加密将会提高系统运行的速度;基于各个厂商提供的 SOAP 解析方式不尽相同,建立统一的符合 SOAP 消息规范的 SOAP 消息也是一个重要的方面。我们也可以对系统进行进一步的扩展。如果在服务器端建立一个内部的 CA,服务提供商为每一个客户端应用都建立一个密钥对,同时在服务器端为每一个用户建立私钥和用户注册信息之间的对应。使用密钥管理系统对密钥进行统一管理,这样在用户发出竞价信息经过签名后,在服务器端提供验证功能,就能实现客户端和服务器的双向认证。

参考文献:

- [1] Triple DES Encryption[EB/OL]. <http://www.tropsoft.com/strongenc/des3.htm>, 2004-07.
- [2] 段智华. SOAP 技术及其安全性研究[EB/OL]. <http://www-900.ibm.com/developerWorks/cn/xml/x-soapsec/index.shtml#1>, 2001-07.
- [3] HELTON R, HELTON J. Java 安全解决方案[M]. 袁泉, 吴静, 等译. 北京: 电子工业出版社, 2003.
- [4] X. 509 Certificates and Certificate Revocation Lists[EB/OL]. <http://java.sun.com/j2se/1.4.2/docs/guide/security/cert3.html>, 2001.
- [5] APSHANKAR K, AYALA D. 开放源代码的 Web 服务高级编程[M]. 周辉, 杜一民, 译. 北京: 清华大学出版社, 2002.
- [6] 唐大仕. 用 Java/C# 开发手机程序及移动应用[M]. 北京: 电子工业出版社, 2004.
- [7] [美] SEELY S. SOAP: XML 跨平台 Web Service 开发技术[M]. 杨涛, 等译. 北京: 机械工业出版社出版, 2004.
- [8] 程尊平, 等. 基于 SOAP 的 XML 安全模式研究[J]. 计算机科学, 2003, 30(10): 162-165.