中国科学院 SAMP 系统的加密通信①

肖 欧[1,2], 尹震宇[2]

[1](中国科学院大学, 北京 100049)
[2](中国科学院 沈阳计算技术研究所, 沈阳 110168)

摘 要: 中科院仪器设备共享管理平台(Apparatus and Equipment Sharing Management Platform of Chinese Academy of Sciences, 以下简称 SAMP)在通信过程中, 存在重要数据在传输时没有加密、可能会接收到第三方恶意数据等数据通信安全问题, 导致 SAMP 系统可能存在数据被窃取、收到的数据不安全等威胁. 针对这些问题, 结合 SAMP 系统的特点以及其通信数据的特征, 将基于 Axis2 Rampart 模块的 WS-Security 数字签名和加密技术应用于 SAMP 数据通信安全问题. 本文首先简单介绍了 Axis2 框架、数字签名和加密技术、Axis2 Rampart 模块加密原理, 然后基于 Axis2 Rampart 模块, 采用数字签名 + 口令 + 非对称加密的 WS-Security 通信安全方案, 对 SAMP 系统网络之间的数据传输接口进行封装, 实现了消息传输的签名、认证和加密、解密过程, 最后测试并分析对比了使用安全方案前和使用安全方案后, SAMP 的 Web Services 数据传输接口的响应处理时间和 CPU 占用率. 实验结果表明: 使用 Axis2 Rampart 模块 + 数字签名 + 口令 + 加密来保证 SAMP 系统的数据通信安全问题具有高安全性、高可扩展性和高响应处理速度, 符合实际应用需求, 也能够广泛的推广到其他企业 Web 应用中去.

关键词: WS-Security; Axis2; Rampart; 签名与加密; 模块扩展; SAMP

Communication Encryption of Axis2 in Apparatus and Equipment Sharing Management System of Chinese Academy of Sciences

XIAO Ou[1,2], YIN Zhen-Yu[2]

[1](University of Chinese Academy of Science, Beijing 100049, China)
[2](Shenyang Institute of Computing Technology, Chinese Academy of Science, Shenyang 110168, China)

Abstract: Considering these security problems of data transmission such as some import data is transited without encryption, malicious data might be received from a third part in Apparatus And Equipment Sharing Management Platform of Chinese Academy of Sciences (SAMP), SAMP system may result these threats of data theft and data receive insecurity. To solve these problems, combined with the characteristics of SAMP System and its features of data transmission, the technologies of digital signature and encryption of WS-Security based on the Axis2 Rampart module are applied to the issue of data communication security of SAMP System. Firstly, the Axis2 framework, digital signature, encryption technology and the encryption principle of Axis Rampart module are introduced by this paper briefly. After that, encapsulate the data transmission interface of SAMP System by using a data transmission security solution, which uses a combination of digital signature + password + asymmetric encryption of WS-Security. It implements the signature, authentication, encryption and decryption process of message transmission. Finally, it also tests and analyzes the processing time of the response and CPU usage of data transmission interface in SAMP System. The experiment results show that by using the combination technology of Axis2 Rampart module + digital signature + password + encrypt, it can ensure that the security issue of data transmission of SAMP system has a high security, high scalability and high response speed, meet the demand of practical application, and be widely promoted to other Web Application of Enterprise.

Key words: WS-Security specification; Axis2; Rampart; signature and encryption; modular extension; SAMP

① 基金项目:“数控系统功能安全技术研究”国家科技重大专项(2014ZX04009031)

收稿时间:2015-09-18;收到修改稿时间:2015-10-26

中科院仪器设备共享管理系统是一个分布式系统, 主要包括院中心服务器和刷卡系统. 院中心服务器由应用服务器集群、数据库集群和负载均衡服务器组成, 部署在外网环境, 供研究人员预约设备、管理人员管理设备、管理预约及查询统计等使用. 刷卡系统作为仪器设备共享管理系统的子系统部署在各个研究所, 刷卡系统主要包括刷卡服务器、刷卡器. 每个研究所的仪器使用人员通过刷卡使用仪器, 并将仪器使用时间、开关机时间、仪器温度、仪器湿度等传送给该研究所的刷卡服务器进行保存. 刷卡服务器实时推送刷卡信息、仪器使用信息到中心应用服务器, 供管理人员查询、管理仪器. 同时, 刷卡系统还需要对刷卡上机的用户进行验证和权限控制, 以保障系统的安全性, 因此刷卡服务器还需要定时调用院中心应用服务器提供的 Web Service 服务接口, 及时获取新的用户信息、仪器信息、仪器授权信息、仪器管理员信息等.

截止 2013 年, “中科院仪器设备共享管理系统”入网设备达到 4512 台套, 其中共享设备达到 4089 台套. 系统处理入网仪器委托单数 51 万个, 其中共享仪器委托单数 49 万个. 中科院仪器设备共享管理系统整体架构图如图 1 所示.

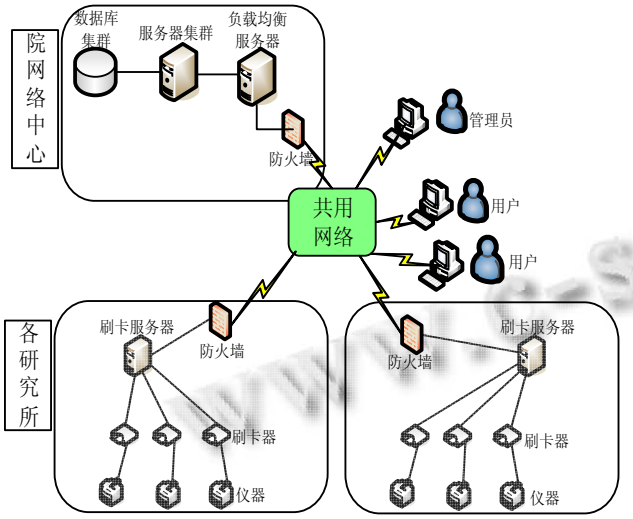

图 1　仪器设备共享管理系统整体架构图

院中心服务器与各研究所的刷卡服务器基于 Web Service 进行数据通信. 目前, 已经投入使用的中科院仪器设备共享管理系统 V2.0 没有考虑到中心服务器与刷卡服务器进行数据传输时的数据通信安全方面的问题. 通过对中科院仪器设备共享管理平台的数据通信过程进行分析, 中科院仪器设备共享管理系统的中心服务器与所级刷卡服务器在进行数据通信时可能存在以下数据安全问题:

① 中心服务器的数据库中存储的用户密码等重要信息在传输时被他人截获;

② 刷卡服务器接收到第三方发来的非法数据, 并将其存入了数据库;

③ 通信双方不能确保传输的数据在传输过程中没有被他人篡改.

这些安全问题可能导致中科院仪器设备共享管理系统存在数据可能被窃取、收到的数据不安全等威胁. 随着中科院仪器设备共享管理系统 V3.0 中正被逐渐投入使用的仪器、设备越来越多, 重要数据的数据量也随之增加, 解决这些安全问题已刻不容缓. 另外, 由于刷卡服务器与中心服务器之间进行定时数据通信时, 消息响应数据量比较大, 因此在解决上述存在的安全问题的同时, 添加安全策略后的中科院仪器设备共享管理系统的数据通信性能也是一个很重要的关注点.

IOASIS 组织于 2006 年 2 月 17 日发布了一个关于 Web 服务安全性(Web Services Security, WS-Security) 的 WS-Security 标准, 该规范描述如何向 SOAP 消息附加签名和加密报头[1]; 针对不同等级的安全需求, 可以使用不同的方法来确保安全性, 以下是当今使用最普遍的几种安全手段:

① J2EE Web 应用默认的访问控制;

② 使用 Axis 的 Handler 进行访问控制;

③ 使用 Servlet Filter 进行访问控制;

④ 使用 SSL/HTTPS 协议来传输;

⑤ 采用 WS 安全规范对消息处理后再传输.

以上前四种措施, 对于那些对安全性需求不是很高的系统是没问题的. 他们使用 Web 认证技术来进行安全保障, 从而确保系统资源不被非法访问. 但是, 即使他们对身份做了认证, 但是消息还是以 plain text 的方式传输的, 还是存在被他人窃取的安全问题. 采用最后的 WS 安全规范, 报文被处理后(签名+加密), 再进行传输, 可以确保即使被第三方获取, 也不能得到解密后的内容. 因此, 对于中科院仪器设备共享管理系统这样要求高安全性的应用, 采用 WS-Security 来保障 Web 服务的安全是很有必要的.

本文在公共密钥加密、数字签名、数字认证、密

钥和非对称加密等加密技术基础上仔细研究了 Axis2 的消息处理机制[2], 结合 Axis2 开源框架和 Rampart 安全模块, 使用 RSA 加密算法、口令保护(保护私钥安全)为基础, 使用“数字签名+数据加密+口令保护”, WS-Security 安全策略[3]来达到中科院仪器设备管理系统的安全性基本要求. 最后对比分析了加密的 SOAP 报文和未加密的 SOAP 报文、签名和加密过程占整个数据通信过程的百分比、不同加密数据量时的响应消息处理速度以及 CPU 占用率来分析出使用 Axis2+Rampart 安全模块来确保中科院仪器设备共享管理系统数据通信安全的必要性、可靠性、可行性.

1 Axis2

Axis 框架来自 Apache 开放源代码组织, 它是基于 JAVA 语言的最新的 SOAP 规范(SOAP 1.2)和 SOAP with Attachments 规范(来自 Apache Group)的开放源代码实现. 模块化是 Axis2 引进的新概念, 它为 Axis2 引擎提供了扩展机制. 它是经过精心设计, 支援轻松添加插件"模组 module", 以提升现有的功能特征, 例如安全性和可靠性的系统[1]. Axis2 体系结构图如图 2[3]所示. 整个 Axis2 包括以下几个[1]:

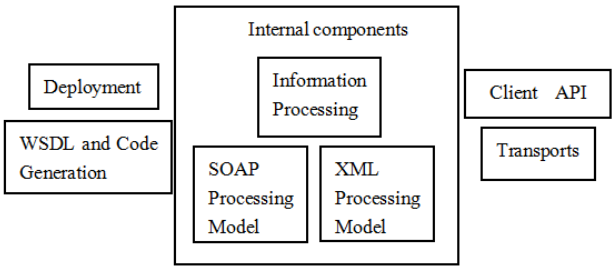

图 2 Axis2 体系结构图

(1) 消息流子系统

它有着灵活的消息传递框架, 这个框架主要包含处理程序, 处理程序被组合成为链, 可以使用一个部署描述符, 来配置处理程序的顺序.

(2) 传输框架子系统

Axis 提供了一个传输框架, 通过这个传输框架, 用户创建自己的可插式传输发送器, 传输侦听器.

(3) 数据编码子系统

Axis 完全遵循 XML Schema 规范, 保证了自动系列化各种数据类型, 并且对于定制的序列化, 反序列化器, 也提供扩展接口来使用它们.

Axis2 引擎由输入流(In-Flow)和输出流(Out-Flow)两个流组成[1], 如图 3[1]所示. 当 Axis2 Engine 接收到消息时, Input Flow 将被调用. 而当它发出消息时, Output Flow 将被调用. 由下图可知, 输入输出流由一些 phase 组成. 这些相关的 phase 按一定的顺序组织, 被放在了一个一维容器中. 其实, phase 也是逻辑上的概念. 每个 phase 是由一系列关联的类组成的. 最终形成一个处理链, 来完成某个逻辑功能. 在 Axis2 引擎中, 处理类是最小的逻辑单元. 当一个 phase 被调用时, Axis2 将依次调用组成该 phase 的处理类.

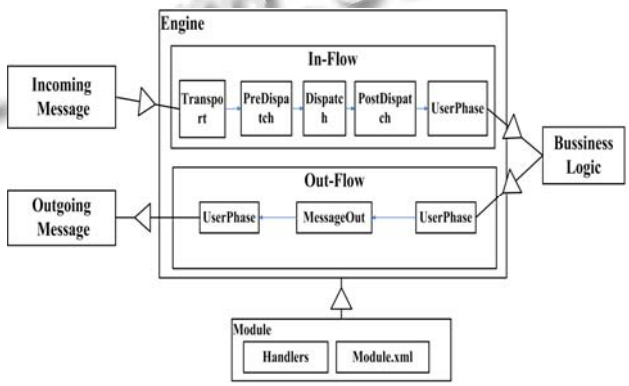

图 3 Axis2 引擎

2 加密解密基本概念

2.1 非对称加密

这个是最重要的概念之一. 非对称加密, 它有一把公钥, 一把私钥. 公钥能够传给其他人, 一般称为 public key. 经过传播后, 一系列相同的公钥对应唯一的私钥. 私钥是自己私有的, 不可以传给别人, 一把私钥可以有多把公钥. 非对称加密过程为: 公钥加密, 私钥解密, 整个过程如图 4[1]所示.

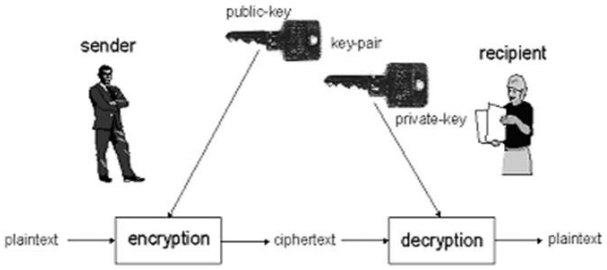

图 4 非对称加密过程

2.2 数字签名

我们已经了解到, 私钥只能被一方持有, 而公钥可以被多方持有. 数字签名, 即发送方对要发送的报

文做如下处理: 先用 Hash 函数进行 hash, 产生 digest, 接下来用自己的私钥, 加密这个 digest, 生成“数字签名”消息, 结合报文, 发送给报文接收方. 接收方收到后, 获取数字签名, 使用之前发送方传的公钥, 对其解密, 得到摘要. 通过摘要, 可以确定消息是由对方发出的. 最后接收方再对报文进行 Hash, 将得到的结果与上一步得到的摘要进行对比. 如果两者一致, 则说明消息未被修改过. 这个过程可以总结为“私钥加密, 公钥解密”. 数字签名流程如图 5 所示.

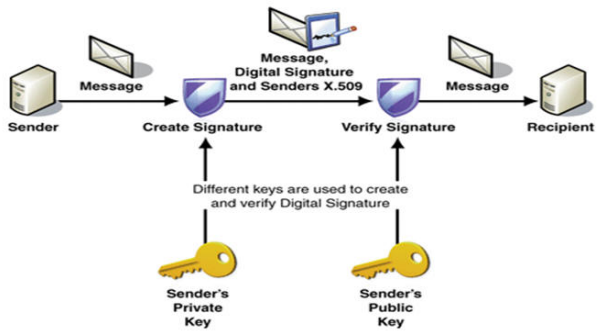

图 5　数字签名流程

2.3 口令保护

在非对称加密算法中, 一把私钥与多把公钥对应. 如果 private key 被丢失或者被窃取, 那么数据传输也不能保证是安全的. 因此在 private key 上加一道锁来保证私钥的安全也是很有必要的, 口令保护过程如图 6 所示.

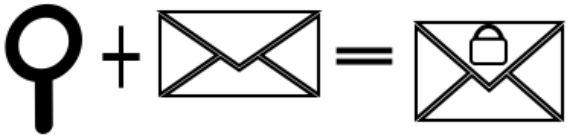

图 6　口令保护

从图 6 可知, 要得到私钥, 就要先有一个解开私钥信封的密码, 一般称之为“口令”. 在没有得到这个口令时, 即使密钥被他人获取, 也不能进行解密, 也就不会造成安全威胁.

3　Axis2 Rampart模块

3.1 rampart 工作原理

Web Service 安全规范, 主要是为了确保在 client 端和 service 端进行消息传递的安全, 而制定的如何进行加密、认证的方式. 在 Axis2 中, 是以 handler 的模式来实现 web service 安全规范的. 在旧的 Axis 版本中, 要实现 client 端和 service 端之间的安全通信, 即双方之间的 web service 认证和加密, 使用者需要自己实现 handler. 在 Axis2 1.4 版以及后续版中, Axis2 为我们提供了 rampart[1], 这个 handler 不再需要我们写了. rampart 是 axis2 实现 ws-security 的一个必须模块, 主要是基于 WSS4J 来保证任务安全.

Rampart 是对 handler 的一种封装, 主要基于 Handler 模式, 类似于一个拦截器. Handler 模式如图 7 所示. 基于 Rampart, 只需要修改客户端与服务器端的 XML, 就可自由在不同的 web service 安全规范之间切换. 因此只要应用 Rampart, 我们只要有“用户名”、口令, 并提供服务器端和客户端的 jks(一种文件, 保存自己的私钥和对方的公钥), Rampart 就可以自动为我们进行加密、解密、认证功能.

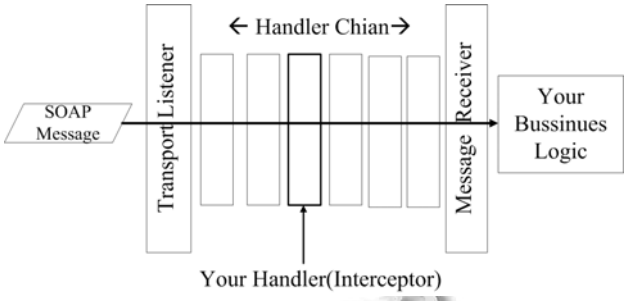

图 7　Handler 模式

3.2 Axis2 WS-Security 数字签名和加密过程

实现客户端访问一个 web service, 彼此间只通过 http, 因此需要对客户端传给服务端的 message 进行加密, 这时使用的是服务端的公钥, 同时, 客户端使用自己的私钥对 message 进行数字签名. 服务器的服务端得到客户端传来的加密的 message 后, 首先使用客户端的公钥进行认证, 如果通过, 则使用自己的私钥解密并把要返回给客户端的消息进行同客户端一样的签名、加密操作返回给客户端. 客户端得到服务端返回后的被加密的消息后进行认证、解密, 并显示在客户端, 具体过程如图 8 所示.

3.3 使用 Rampart 模块进行签名和加密的配置、实现过程

使用 Rampart 对 SOAP 消息进行签名和加密, 主要分为以下几个步骤[1], 详细实现请查看参考文献 1:

(1) 使用 keytool[4]生成密钥仓库, service.jks 和

client.jks, 这两个文件分别存放自己的私钥和对方的公钥, 根据 2.2 节的“数字签名”使用“私钥签名, 公钥认证”可以实现对 SOAP 消息进行签名和认证, 而根据 2.1 节的“非对称密钥”使用“公钥加密, 私钥解密”可以实现对 SOAP 消息进行加密和解密, keytool 的具体使用情况请看参考文献 4. 密钥文件及私钥加密口令如表 1.

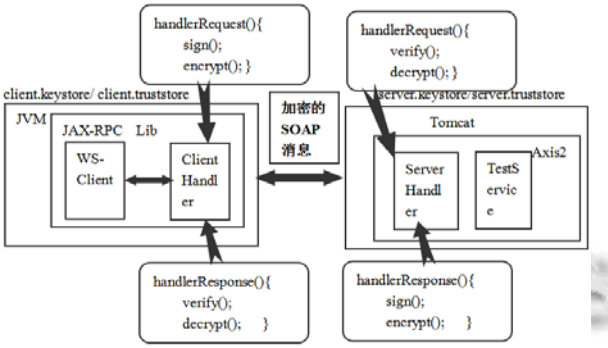

图 8 SOAP 消息签名和加密过程

表 1 密钥文件及私钥加密口令

	口令	File Name
Client 端	SAMPclient	SAMPclient.jks
Service 端	SAMPserver	SAMPservice.jks

(2) 配置 service 端和客户端进行加解密需要的配置文件 SAMPservice.properties, SAMPclient.properties. 这里只列出 service 端配置文件 SAMPservice.properties, 客户端配置文件 SAMPclient.propertiesde 的配置与 service 端类似.

SAMPservice.properties

```
org.apache.ws.security.crypto.provider=org.apache.ws.security.components.crypto.Merlin
org.apache.ws.security.crypto.merlin.keystore.type=jks
org.apache.ws.security.crypto.merlin.keystore.password=serverSAMP
org.apache.ws.security.crypto.merlin.file=SAMPservice.jks
```

(3) 配置 Rampart, 主要配置 client 端描述文件 axis2.xml, 服务端描述文件 service.xml. 具体配置含义请参考关于 Axis2 的 Rampart WS-Security 模块配置主页[10]. axis2.xml 如下, service.xml 配置类似.

axis2.xml

```
<module ref="rampart"/>
<parameter name="OutflowSecurity"><action>
    <items>Timestamp Signature Encrypt</items>
    <user>SAMPclient</user> <!-- 使用 keytool 生成 jks 文件时, private key 的 alias -->
    <signaturePropFile>SAMPclient.properties
    </signaturePropFile>
    <passwordCallbackClass>
        com.SAMP.axis2.security.SAMPPasswordCB
    </passwordCallbackClass>
    <signatureKeyIdentifier>
        DirectReference
    </signatureKeyIdentifier>
    <encryptionIdentifier>
        SKIKeyIdentifier
    </encryptionIdentifier>
    <encryptionUser>SAMPserver</encryptionUser>
  </action></parameter>
<parameter name="InflowSecurity">
  <action>
    <items>Timestamp Signature Encrypt</items>
    <passwordCallbackClass>
        com.SAMP.axis2.security.SAMPPasswordCB
    </passwordCallbackClass>
    <signaturePropFile>
        Client.properties
    </signaturePropFile>
  </action>
</parameter>
```

(4) 编写密码回调类 SAMPPWCBHandler, 使用该类来得到签名、认证和加解密需要的口令, 回调类实现代码如下:

```
public void handle(Callback[] callbacks) throws IOException,UnsupportedCallbackException {
  for (int i = 0; i < callbacks.length; i++) {
    WSPasswordCallback pwcb =
        (WSPasswordCallback) callbacks[i];
        String identifier = pwcb.getIdentifier();
    if ("SAMPclient".equals(identifier)) {
        pwcb.setPassword("clientSAMP");
    } else if ("SAMPserver".equals(identifier)) {
        pwcb.setPassword("serverSAMP");
    }
  }
}
```

(5) 在 service 端(中心服务器)添加 SAMP 中需要加密传输的数据通信服务接口, 并在 services.xml 中配置, 发布服务. 仪器设备共享管理系统中各所级刷卡服务器与中心服务器进行交互数据时, 需要保密的关键数据都是由刷卡服务器定时调用中心服务器提供的 Web Services 服务, 因此中心服务器作为 serivce 端, 而所级刷卡服务器则作为客户端. 在中心服务器的 AxisSAMPService 类中添加以下四个服务:

a.用户信息同步接口

public string synUserInfo(String authStr, String commandStr, String bodyStr)

b.仪器授权信息同步接口

public string synApparatusGrantInfo(String authStr, String commandStr, String bodyStr)

c.仪器管理员信息同步接口

public string synApparatusAdminInfo(String authStr, String commandStr, String bodyStr)

b.仪器信息同步接口

public string synApparatusInfo(String authStr, String commandStr, String bodyStr)

客户端在请求这四个服务接口时, Axis2 会根据 services.xml 的配置, 先对请求参数进行认证、解密, 获得真实的参数, 根据参数, 调用仪器设备共享管理系统的底层业务代码, 得到响应消息后, 进行签名、加密后返回给客户端.

(6) 编写客户端(所级刷卡服务器)调用相应的服务, 获得返回结果. 客户端在请求 service 端服务时, 会根据 axis2.xml 的配置, 先对请求参数进行签名+加密, 得到返回结果时, 也会先进行认证、解密, 才得到最终结果.

最后, 对已经配置好的应用进行不同的实验测试, 得到结果, 分析使用 Axis2 + Rampart+口令+签名和加密时进行数据通信以及没有使用签名和加密进行数据通信时消息响应处理时间、CPU 占用率等情况.

4 对签名和加密的性能进行实验测试与结果分析

4.1 不使用加密和签名的 SOAP 消息与使用了加密和签名的 SOAP 消息结果对比

(1) 在没有对 Rampart 模块进行配置时, 使用 SOAPMonitor 监视器得到的客户端访问服务端的请求消息和响应消息如表 2 所示.

表 2 没有签名和加密的 SOAP 消息

客户端请求消息	<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"><soapenv:Body> <axis2ns2:synUserInfo xmlns:axis2ns2="http://service"> <authStr>{"userName": "zs", "password": "2343111", "sessionId": "1422940727395"}</authStr> <commandStr>{"command":"byId"} </commandStr> <bodyStr>{"apparatusId":"2314"} </bodyStr> </axis2ns2:synUserInfo> </soapenv:Body></soapenv:Envelope>
服务端响应消息	<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"> <soapenv:Body> <ns:synUserInfoResponse xmlns:ns="http://service"> <ns:return> {"status":"0","description":"succesus", "sessionId": "1422940727395", "resultSet": [{ "userId": "9653", "userName": "病理分析郝俊峰", "type": "5", "belongUnitName": "生物物理研究所","belongUnitNo": "153311", "belongUnitId": "58", "belongGroupName": "病理分析组", "belongGroupNo": "103", "telephone": "64888424", "email": "haojf@moon.ibp.ac.cn", "cardNo": "2171795964", "invalidDate": "2015-12-31", "userValidFlag": "1"}]} </ns:return> </ns:synUserInfoResponse> </soapenv:Body> </soapenv:Envelope>

从表 2 可以看到, 请求消息和响应消息都是以明文传输的, 如果一些重要的数据也直接这样传输, 被

第三方截获的话, 是可以直接获取或者篡改的.

(2) 使用 3.3 节介绍的步骤对 Rampart 进行相应的配置后, 这里 keytool 在生成 JKS 文件时, 使用 RSA 算法生成公钥和私钥. 在已经配置好 axis2.xml 的客户机中, 使用相同的 client 端代码访问使用 service 端. SOAPMonitor 监视器得到的客户端发往服务器的请求消息主要部分如下:

SOAP 安全头部

```
<wsse:Security>
  <wsu:Timestamp>
    <wsu:Created>
      2015-04-30T07:16:02.961Z
    </wsu:Created>
    <wsu:Expires>
      2015-04-30T07:21:02.961Z
    </wsu:Expires>
  </wsu:Timestamp>
  <ds:Signature>
   <ds:SignatureMethod     Algorithm=
      "http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
   <ds:SignatureValue>
      uQz+S9JA2Lo1Pay2HNrfFdSzXp5Z......TZSsx0=
  </ds:SignatureValue>
  <ds:DigestMethod     Algorithm=
      "http://www.w3.org/2000/09/xmldsig#sha1"/>
  <ds:DigestValue>
      ow1sDdh3ec306e2r9EP7ukQaqvA=
  </ds:DigestValue>
</ds:Signature>
<xenc:EncryptedKey>
<xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5" />
<xenc:CipherValue>
    cO6kfBWWaTZF8Z4FKsaI......PDESULg=
</xenc:CipherValue>
</xenc:EncryptedKey>
</wsse:Security>
```

SOAP 消息中的<wsse:Security>头部非常重要. 它保存着所有运行时的安全配置信息, 如签名数据、使用的签名、加密算法、加密了的密钥.

SOAP Body 内容

```
<xenc:EncryptedData >
      <xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"
/><xenc:CipherData>
      <xenc:CipherValue>YMaW+eKzG6uOn+3MfRfDGmOoTp5..
.lgzQ0=</xenc:CipherValue>
      </xenc:CipherData>
</xenc:EncryptedData>
```

<xenc:CipherData>元素为加密数据. 这个已加密的数据, 从安全头部引用<xenc:EncryptedKey>?值, 作为在 Body 内容中, 用于对称加密的密匙.

服务器对客户端的请求消息, 主要处理流程是: 先使用 client 端的公钥进行认证, 再使用自己的私钥进行解密, 得到请求参数, 构造出响应消息后, 对响应消息进行先签名, 后加密, 再发送给客户端.

对比(1)中的使用明文传输, 没有经过任何签名和加密等 Web 安全技术进行处理的 SOAP 消息和(2)中使用了 Rampart 签名和加密技术进行处理了的 SOAP 消息, 很容易得出, 与明文传输相比, 使用签名和加密处理过的 SOAP 传输消息安全的多. 其次, 在签名和加密时, 使用了 RSA 算法[3]. 这种算法的特点: 密钥越长, 它就越难破解. 目前被破解的, 最长的 RSA 密钥, 是 768 个二进制位. 换句话说, 对于超过 768 位的密钥, 人类还无法破解. 可以这么认为, 1024 位的 RSA 密钥, 是基本安全的. 而 2048 位的密钥, 则极其安全. 而本文在使用 keytool 生成公钥和私钥时, 使用的 1024 位 RSA, 所以可以说达到了前面提到的安全性基本要求.

4.2 不使用加密和签名时与使用了加密和签名时, 响应消息处理时间与 CPU 占用率结果对比

(1) 不使用 Axis2 + Rampart WS-Security 进行签名和加密时和使用了 Axis2+Rampart WS-Security 进行签名和加密时的消息响应处理时间结果对比

在 client 端调用 service 端的四个需要加密传输数据的接口时, 需要在 client 端调用服务接口的那行代码前插入如下一行代码:

long startTime = System.currentTimeMillis();

在调用服务接口的那行代码后面插入如下一行代码: long endTime = Systemj.currentTimeMillis();

这样就可以得到消息从请求到响应的处理时间:

endTime - startTime 了.

表3显示了不使用WS-Security进行签名和加密时和使用WS-Security进行签名和加密时，调用四个服务接口时的消息处理时间. 这里明文传输和签名+加密传输时，接口的请求参数和响应消息都是一样的. 接口的请求参数大小都为30bytes左右，消息响应大小都为 1M 左右. 测试 3000 次，取消息响应处理时间的平均值.

表 3　明文与加密时消息响应平均时间(ms)对比

Web Service 通信接口	明文	签名+加密	差值
synUserInfo	344	421	77
synApparatusInfo	329	422	93
synApparatusGrantInfo	327	390	63
synApparatusAdminInfo	312	391	79m

表 3 的测试结果中，这里的差值从表面看，指的是签名+加密的消息响应处理时间-明文的消息响应处理时间. 测试是在局域网中进行的测试，这个测试环境比较纯粹，可以忽略掉消息在网络中的通信时间. 因此这个差值完全可以被当作是使用 Axis2 Rampart 模块进行签名、加密的处理时间. 通过计算，可以得出四个接口中，签名+加密时间占整个消息响应处理时间的百分比如表 4 所示.

表 4　签名+加密时间占整个消息响应时间百分比

SAMP 通信接口	数据加密所占百分比(%)
synUserInfo	18
synApparatusInfo	22
synApparatusGrantInfo	16
synApparatusAdminInfo	20%

从以上的四个接口签名+加密时间占整个消息响应处理时间的百分比，可以得出签名+加密时间占整个消息响应处理时间的百分比范围为 16%~20%. 因此对于同样的请求参数和响应消息，使用 Axis2 Rampart 模块+WS-Security 签名、加密策略与不使用 WS-Security 时，对消息的响应处理时间的影响最多也只是增加 20%. 而在中科院仪器设备共享管理系统中，刷卡服务器设定为每 1 小时请求一次中心服务器的四个接口，来获得用户信息、仪器授权信息等信息. 所以在中科院仪器设备共享管理系统中增加了 Axis2 Rampart 模块+WS-Security 签名、加密模块后所增加的消息响应处理时间对整个数据通信来说是很小的. 以增加较小的消息签名+加密处理时间来换取整个 SAMP 系统的数据通信安全是完全有必要的.

(2) 使用 Axis2 + Rampart WS-Security 进行签名和加密，对比大数据量的响应消息和小数据量的响应消息时消息响应处理时间结果

表 5 显示了使用 WS-Security 进行签名和加密时，调用四个服务接口，对每个接口都使用 7 组测试序列进行测试. 第一组的消息响应大小为 1M，后面每组的消息响应大小依次为前一组的 5 倍. 之所以划分为 7 组数据，是因为中科院仪器设备共享管理系统中，中心服务器跟刷卡服务器进行数据通信时，最大的消息响应大小为 30M，且每组的大小是其前一组的 5 倍，这个划分足够细，而且能够便于测试. 这七组测试的请求参数大小都是一样的. 这样可以对比小数据量的响应消息和大数据量的响应消息时，使用 Axis2 Rampart 模块的消息处理速度. 这里总共进行 3000 次测试，取消息响应处理时间的平均值.

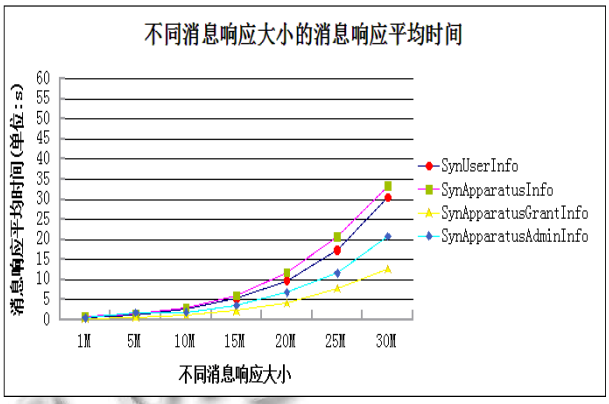

图 9　不同数据量响应消息的消息响应平均时间

对比图 9 的测试结果，可以得出使用 WS-Security 进行签名和加密，调用 4 个接口的请求参数一样时，小数据量的响应消息和大数据量的响应消息所用的消息处理时间并不是与数据量的增长成正比的，当消息响应大小增加 5 倍时，消息响应处理时间最大也只是增加 2 倍左右. 针对与中科院仪器设备管理系统需要保密传输的响应数据最大也不超过 30M 这个特性，使用 Axis2 + rampart 模块的 WS-Security 签名+加密方式处理大数据量的消息响应的消息处理时间最大也只为 1 分钟左右，而刷卡服务器设定为每 1 小时请求一次中心服务器的四个接口，这个响应时间完全在可接收的范围内.

(3) 不使用 Axis2 + Rampart WS-Security 进行签

名和加密时和使用了 Axis2+Rampart WS-Security 进行签名和加密时的 CPU 占用率结果对比

为了测试使用 Axis2+Rampart WS-Security 进行签名和加密时的性能开销, 本文采用了以下方法:

在没有使用 WS-Security 进行加密传输时, 使用 client 端调用 service 端提供的四个服务. 使用 JDK bin 目录下自带的 jvisualvm 工具对当前 client 端和 service 端应用程序的 cpu 占用率进行记录. 之后使用 WS-Security 进行加密传输, 同样记录 client 端和 service 端应用程序的 cpu 占用率, 对比分析使用 Axis2 + Rampart WS-Security 进行签名和加密时的性能开销. 这里使用的消息响应数据量大小为 10M, 并进行多次测试, 取 CPU 占用率的平均值.

表 5 显示了使用 jvisualvm 工具记录的没有使用 WS-Security 时的 client 端 和 service 端 CPU 占用率以及 使用了 WS-Security 的 client 端和 service 端的 CPU 占用率. 测试 2000 次, 取 CPU 占用率的平均值.

表 5 明文与加密时 CPU 平均占用率(%)对比

	明文	签名+加密	差值
client 端	35.4	42.3	6.9
service 端	36.8	47.5	10.7

对比表 5 中, client 端在使用 WS-Security 与不使用 WS-Security 传输数据时的 CPU 占用率, 可以得出使用 Axis2 + Rampart 模块的 WS-Security 签名和加密安全策略, 并不会占用很多 CPU 资源. 同理, 对比 serivce 端在明文和签名+加密两种方式下的 CPU 占用率, 也能得出同样的结果. 因此牺牲最多 10%左右的 CPU 开销来保证中科院仪器设备共享管理系统的数据传输安全是非常值得的.

5 结论

对于需要支持跨编程语言, 要求高安全性、高可扩展性的中科院仪器设备共享管理系统来说, 基于 Axis2 的 Rampart 模块, 使用 Header 和 Body 的 WS-Security 签名, 使用 timestamp 和主体加密, 同时对密钥进行加密的安全策略来保证 Web Service 的数据通信安全性问题是目前为止最合适的选择. 另外, 根据本文的实验测试可以知道, Axis2 在基本请求/响应处理方面, 性能开销不是很大, 同时使用 Axis2 的 rampart 模块来支持数据通信的签名+加密功能, 具有高可扩展性[5], 完全能够满足中科院仪器设备共享管理系统对于数据通信安全的要求. SOAP 消息实验结果和性能测试数据表明: 使用头部和主体 WS-Security 签名 + 时间戳 (防重播攻击) + 密钥加密(口令) + 主体加密安全策略, 并根据实际情况结合相应的 Web Service 开源框架进行 Web 服务安全性保证是可行的, 且对推动 Web Service 框架在企业级应用上的使用具有很重要的实际意义.

参考文献

1 Rosenberg J. Securing Web Services with WS-Security: Demystifying WS-Security, WS-Policy, SAML, XML Signature, and XML Encryption. New York: Sams Publishing, 2004: 245−304.

2 Gao L, Liu SF. A solution of Axis2 message routing and Web Services Security. IEEE International Conference on Web Service. 2011. 384−388.

3 马安峰.基于 Rampart 模块的 Axis2 Web 服务安全研究.计算机应用与软件,2009,26(9):31−34.

4 梁栋.Java 加密与解密的艺术.北京:机械工业版社,2014: 105−152.

5 邓子云,黄婧,罗涛.XFire 和 WSS4J 下的 Web Service 组合安全策略.计算机系统应用,2009,18(9):53−56.