XML 加密技术研究及其在 Java 中的实现

陈志涛　邬家炜

(华南师范大学计算机科学系　广州 510631)

摘　要　在比较各种现有的加密技术的基础上,阐述 XML 加密的原理,指出该加密标准的优越性,并通过示例解释 XML 加密的方法,最后给出了一个基于 Java 的 XML 加密的具体实现。

关键词　XML,XML 加密,PKI,信息安全,Java

1 引言

可扩展标记语言(XML)[1]作为一种 Internet 上的信息交换格式,其普及性仍然在增长,许多新兴的技术,比如 Web 服务,都将 XML 广泛应用于信息交换。而与信息交换有关的一个重要问题是安全。没有保证信息的安全性和可靠性的机制,任何信息交换格式都是不完整的。因此,XML 在传输和存储时的安全性成为非常重要的问题。针对 XML 数据传输存储的安全需求,世界互联网联盟(W3C)发布了 XML 加密规范[2]。

本文首先叙述了 XML 加密的原理,比较了 XML 加密技术与当前现有的加密技术异同点,然后阐述了 XML 加密方法,最后给出了一个利用 Verisign 的 Trust Services Integration Kit(TSIK)[3] 实现 XML 加密的 Java 实例。将 XML 语言引入加密技术后,打破了传统加密技术的局限性,给使用加密的应用程序增加了灵活性和可扩展性。

2 XML 加密原理

XML 加密技术与当前许多现有的加密技术,比如 Netscape 设计的安全套接字层(Secure Sockets Layer,SSL)以及国际工程任务小组(IETF)设计的传输层安全性(Transport Layer Security,TLS)相比,没有本质上的不同,同样基于目前被广泛使用的公共密钥体系(Public Key Infrastructure,PKI)。消息的发送方利用标准的对称加密算法例如 DES、AES(此密钥称为共享密钥)对要传输的原始数据加密,然后利用标准的非对称算法,比如 RSA,用对方的公有密钥对共享密钥进行加密后传送。接受方利用其私有密钥对传送过来的已加密的密钥进行解密,得到共享密钥,然后利用此共享密钥对已加密的数据进行解密,得到原始数据。

现有的加密技术如 SSL、TLS 为通信双方提供了端到端的安全性会话,然而它们却存在一些局限性,比如加密的粒度过粗,只能对整个文档进行加密。基于 HTTP 的 SSL 技术,消息在传送中的每一跳,整条消息都会被加密;消息到达第一个目的地后,整条消息又被解密,在它被重新全部加密传输到第二跳的节点之前,有可能受到嗅探器的威胁。基于 HTTP 的 SSL 提供的加密仅仅在传输的过程中存在,它是不持久的[4]。

XML 加密技术可以突破以上现有技术的局限性,可以对整个文档进行加密,支持对任意数字内容的加密,包括 XML 文档;确保经过加密的数据不管是在传输过程中还是在存储时,都不能被未经授权的人员访问到,即便是在消息传送中的每一跳,都能维持数据的安全性,即不仅数据正在传输的过程中要保证安全性(这就是 SSL 所作出的保证),当数据在某个特定的节点停留的时候,也要保证其安全性。而且 XML 加密技术利用 XML 语言以及 XPath[5]、XPointer、XSLT 的强大表达能力和扩展能力,可以实现特有的功能,即可以在较细粒度上如对文档的特定部分进行签名,可以从 XML 文档中选出一部分内容进行加密,比如在电子商务中,商家可能只需要知道客户的名称、地址和定购的产品信息,但是无需知道任何正在使用的信用卡的各种详细信息,就像银行不需要知道购买货物的详细信息一样,XML 加密技术可以只加密客户的信用卡信息。

3 XML 加密方法

XML 加密语法的核心元素是 EncryptedData 元素,该元素与 EncryptedKey 元素一起用来将加密密钥从发起方传送到已知的接收方,EncryptedData 是从 EncryptedType 抽象类型派生的。XML 加密方式是极其灵活的,其加密粒度可以根据需求的不同而不同,要加密的数据可以是任意数据、XML 文档、XML 元素或 XML 元素内容;加密数据的结果是一个包含或引用密码数据的 XML 加密元素。当加密元素或元素内容时,EncryptedData 元素替换 XML 文档加密版本中的该元素或内容。当加密的是任意数据时,EncryptedData 元素可能成为新

XML 文档的根,或者可能成为一个子代元素。当加密整个 XML 文档时,EncryptedData 元素可能成为新文档的根。此外,EncryptedData 不能是另一个 EncryptedData 元素的父代或子代元素,但是实际加密的数据可以是包括现有 EncryptedData 或 EncryptedKey 元素的任何内容[6]。

下面通过示例演示了 XML 加密中的加密方法[7]。

a. 要加密的样本 XML 文件

```
<? xml version = "1.0"? >
<purchaseOrder >
  < Order >
    < Item > book < /Item >
    < Id > 123 - 958 - 74598 < /Id >
    < Quantity > 12 < /Quantity >
  < /Order >
  < Payment >
    < CardId > 123654 - 8988889 - 9996874 < /CardId >
    < CardName > visa < /CardName >
    < ValidDate > 12 - 10 - 2004 < /ValidDate >
  < /Payment >
< /purchaseOrder >
```

b. 对整个文档加密

```
<? xml version = '1.0' ? >
< EncryptedData xmlns = ' http://www. w3. org/2001/
04/xmlenc # '
    Type = ' http://www. isi. edu/in - notes/iana/
assignments/media - types/text/xml' >
      < CipherData >
        < CipherValue > A23B45C56 < /CipherValue >
      < /CipherData >
< /EncryptedData >
```

加密前的原始数据是 XML 文档并且 IANA 对 XML 的正式类型定义是 Type = 'http://www.isi.edu/in - notes/iana/assignments/media - types/text/xml'。这作为 Type 属性的值出现。XML 对各种流行的数据格式(如 RTF、PDF 和 JPG)使用 IANA 的类型定义。

c. 对单个元素加密。只加密样本 XML 文件中的一个元素,例如 Payment 元素。

使用 Type = 'http://www.w3.org/2001/04/xmlenc # Element',对单个元素加密。

```
<? xml version = '1.0' ? >
< PurchaseOrder >
  < Order >
    < Item > book < /Item >
    < Id > 123 - 958 - 74598 < /Id >
    < Quantity > 12 < /Quantity >
  < /Order >
    < EncryptedData xmlns = ' http://www. w3. org/
2001/04/xmlenc #
      Type = 'http://www.w3.org/2001/04/xmlenc #
Element' >
      < CipherData >
        < CipherValue > A23B45C564587 < /CipherValue >
      < /CipherData >
    < /EncryptedData >
< /PurchaseOrder >
```

d. 加密元素的内容。只加密样本 XML 文件中元素 CardId 中的内容。

使用 Type = 'http://www.w3.org/2001/04/xmlenc # Content'。

```
<? xml version = '1.0' ? >
< PurchaseOrder >
  < Order >
    < Item > book < /Item >
    < Id > 123 - 958 - 74598 < /Id >
    < Quantity > 12 < /Quantity >
  < /Order >
  < Payment >
    < CardId >
      < EncryptedData Type = 'http://www.w3.org/
2001/04/xmlenc # Content'
          xmlns = ' http://www. w3. org/2001/04/
xmlenc # ' >
        < CipherData >
          < CipherValue > A23B45C564587 < /CipherValue >
        < /CipherData >
      < /EncryptedData > < /CardId >
      < CardName > visa < /CardName >
      < ValidDate > 12 - 10 - 2004 < /CardName >
  < /Payment >
< /PurchaseOrder >
```

e. 加密非 XML 数据　本例通过 XML 加密发送一个 JPEG 文件。按字节序列加密的整个 JPEG 文件将作为 CipherValue 元素的内容出现。请注意,b 示例和本实例中之间唯一的差异:EncryptedData 元素的 Type 属性。本示例包含了 JPEG 格式的 IANA 类型。类似地,可以通过提供 IANA 值加密任何格式。

```
<? xml version = '1.0' ? >
< EncryptedData xmlns = ' http://www. w3. org/2001/
04/xmlenc # '
    Type = ' http://www. isi. edu/in - notes/iana/
assignments/media - types/jpeg' >
      < CipherData >
        < CipherValue > A23B45C56 < /CipherValue >
      < /CipherData >
< /EncryptedData >
```

4 利用 Java 实现 XML 加密过程

为了利用 Java 实现 XML 加密,本文实现过程代码必须为 Java Cryptographic Extension (JCE) 安

装并注册一个密码提供程序[8]。JCE 提供程序必须支持生成公钥和私钥的 RSA 算法。但是在 JDK1.4 中,SunJCE 提供程序不适合生成非对称密钥,因为它没有包含 RSA 算法类型的实现。本文示例代码使用了开放源代码的 JCE 提供程序 ISNetworks,它包括了对 RSA 的支持。

步骤 1:生成密钥对

```
//取得一个实现了 RSA 算法的 KeyPairGenerator 对象
KeyPairGenerator pairgen =
    KeyPairGenerator.getInstance("RSA",
"ISNetworks");
//为 KeyPairGenerator 对象提供密钥大小和一个随机数
SecureRandom random = new SecureRandom();
int KEYSIZE = 1024;
pairgen.initialize(KEYSIZE, random);
KeyPair keyPair = pairgen.generateKeyPair();
```

步骤 2:生成共享密钥

```
//获得一个实现了 Triple - DES 算法的 KeyGenerator
对象。
KeyGenerator kg = KeyGenerator.getInstance("
DESede", "ISNetworks");
//对 KeyGenerator 对象调用 generateKey 方法获取共享
密钥
Key ky = kg.generateKey();
```

这个共享密钥用于加密 XML 内容。共享密钥本身使用上一步生成的公钥加密并嵌入 XML 内容中。

步骤 3:将 XML 转换成 DOM 对象

```
Document doc = null;
DocumentBuilderFactory factory =
    DocumentBuilderFactory.newInstance();
File f = new File(fileName);
DocumentBuilder builder = null;
builder = factory.newDocumentBuilder();
doc = builder.parse(f);
```

步骤 4:获得共享密钥(shared secret)

```
Key dataEncryptionKey = getKey();
```

步骤 5:获得公 - 私密钥对中的公钥,用这个公钥给共享密钥加密

```
KeyPair keyPair = getKeyPair();
Key keyEncryptionKey = keyPair.getPublic();
```

步骤 6:利用一个数据加密密钥、一个密钥加密密钥、与这两个密钥相关联的算法,以及将来包含在输出信息中的密钥信息,根据它们来创建一个 Encryptor 对象。

```
//设置 XML 加密规范中数据加密的密钥类型
AlgorithmType dataEncryptionAlgoType = AlgorithmType.
TRIPLEDES;
//设置 XML 加密规范中密钥加密的密钥类型
AlgorithmType keyEncryptionAlgoType = AlgorithmType.
RSA1-5;
//创建有关密钥的信息
KeyInfo keyInfo = new KeyInfo();
//创建 Encryptor 对象
Encryptor enc = new Encryptor(
        doc,
        dataEncryptionKey,
        dataEncryptionAlgoType,
        keyEncryptionKey,
        keyEncryptionAlgoType,
        keyInfo);
```

利用 Encryptor 对象的构造函数或设置函数时指定的算法必须与密钥相符。它的类在 tsik.jar 中的 com.verisign.xmlenc 这个包中。Encryptor 根据 W3C XML Encryption 规范进行加密,可以指定想要使用哪种加密类型,是 Element 还是 Content。Encryptor 要理解 XPath 表达式,这样才能识别出需要加密的 XML 元素。

步骤 7:调用 Encryptor 对象的 encrypt 或者 encryptInPlace 方法,并将 XPath 作为输入参数传入。XPath 定义了 XML 内部需要进行加密的元素。这个元素的所有子元素,以及 XPath 所指向的属性也都要进行加密[9]。

```
String xPath = "/PurchaseOrderRequest/Payment";
XPath xpath = new XPath(xPath);
enc.encryptInPlace(xpath);
XmlUtil.XmlOnStdOut(doc);
```

encrypt 和 encryptInPlace 两个方法都用传入的共享密钥对 XPath 指定的 XML 元素进行加密,两种方法也都用公钥对共享密钥进行加密,并将加密结果嵌入到 XML 加密后的内容之中。这两种方法的唯一区别在于,encrypt 返回一个全新的 DOM 文档,其中包含加密后的数据,而 encryptInPlace 方法对原有的文档本身进行修改,使其中包含加密后的数据。

结束语 XML 加密在公共网络上的数据安全交换中发挥着越来越重要的作用。XML 加密为需要结构化数据安全交换的应用程序提供了一种端到端安全性,可以满足应用程序对数据交换安全性复杂的需求。同时,采用 Java 语言实现 XML 加密,可以运行在任何支持 Java 的平台环境上,具有很好的开放性和扩展性,可以方便添加各种算法实现和密钥信息的处理。

参考文献

1 Bray T, Maler E, Paoli J, Sperberg - McQueen C M. XML 1.0 (Second Edition). W3C Recommendation, Oct. 2000

2 Eastlake D, Reagle J. XML Encryption Syntax and Processing. W3C Candidate Recommendation, March 2002

3 Verisign Trust Services Integration Kit. ttp://www.xmltrustcenter.org/xmlsig/developer/verisign/index.htm

(下转第 123 页)

件。必要时,需要向客户端提供插件程序以配合整个协议的完成。

4.2 电子支付中间件结构

通信层、电子支付协议支持层、基本业务支持层可以被统一纳入电子支付中间件构架的范围。这样,中间件内部的层次结构也给中间件本身带来了便于扩展、良好的可重用性、灵活性、可维护性及兼容性和适应变化的能力。

电子支付中间件 Internet 之上,它不仅要实现互连,还要实现参与方各部件间的互操作与整合,并向使用者提供方便的使用与开发接口。一种典型的支付模式如图 3 所示。

在通讯方面,中间件构架要支持常用的通信协议和通信服务模式,传输各种数据内容,数据格式翻译、数据加密、数据压缩等。

在中间件构架核心,要进行相应的安全控制、并发控制、可靠性和效率保证、协议流程控制等。

在基本业务支持方面,要能提供基于不同平台的丰富而方便的开发接口,支持流行的开发工具和异构互连接口标准等。

在系统管理和控制方面,解决中间件构架本身的配置、监控、协调,为电子商务应用的易用易管理提供保证。

最后以业务中间件的形式提交给商家、支付网关/银行。

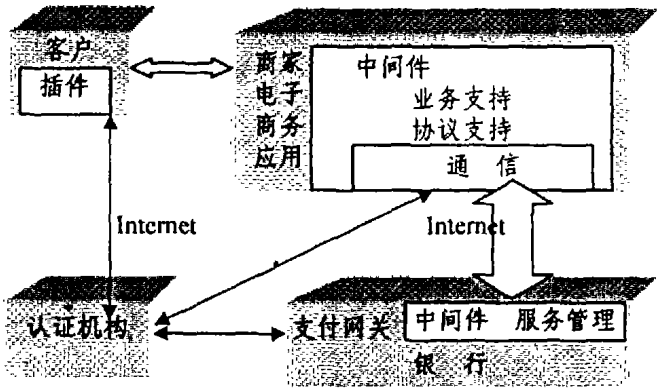

图 3 典型支付模式及支付中间件结构

结束语 目前,安全支付体系的滞后是严重制约电子商务快速发展的重要因素之一。基于层次体系结构的电子支付系统具有良好的扩展性和灵活性强,适应变化的能力强,便于维护。结合中间件直接支持业务逻辑,提高开发效率,增强应用稳定性,和良好的分布式应用支持等特点,用中间件来封装和实现电子支付系统的主要层次,向用户提供强大便捷的应用接口,给应用开发带来了极大的便利,有助于电子支付系统的快速部署和电子商务应用的开展。

参考文献

1 Heikkila J, Laukka M. SPKI Based Solution to Anonymous Payment and Transaction Authorization. http://www.tml.hut.fi /Research / TeSSA/ Papers/Heikkila - Laukka/ nordsec99 - heikkila - laukka.pdf.

2 孙昌爱, 金茂忠, 刘超. 软件体系结构研究综述. 软件学报, 2002,13(7):1228 ~ 1239

3 骆科东, 王建民, 孙家广. 中间件在电子支付中的应用. 计算机工程,2002,8 (28): 240 ~ 243

4 万建成, 卢雷. 软件体系结构的原理、组成与应用. 北京: 科学出版社, 2002

(上接第 117 页)

4 Manish Verma. XML security : Implement security layers, Part 2. http://www - 106. ibm. com/developerworks/xml/ library/x - seclay2. October 21, 2003

5 Clark J, DeRose S. XML Path Language(Xpath) Version 1. 0. W3C Recommendation, Nov. 1999

6 Mactaggart M. Enabling XML security. http://www - 106. ibm. com/developerworks/security/library/s - xmlsec. html, Sep. 2001

7 Siddiqui B. 探索 XML 加密,第 1 部分. http://www - 900. ibm.com/developerWorks/cn/xml/x - encrypt/index.shtml

8 Oaks S . 林琪 译. Java TM 安全(第 2 版). 北京:中国电力出版社, 2002

9 Birbect M ,等. 裴剑锋,高伟,徐继伟,等译. XML 高级编程(第 2 版). 北京:机械工业出版社,2002