基于XSLT的XML安全的应用研究

陈 宁 葛君伟 邓宁波

(重庆邮电学院计算机学院 重庆400065)

摘 要 扩展标记语言(XML)的出现,使得在异构环境下数据的交换和传输成为可能。在应用过程中,对传输中文档安全性的要求就显得异常重要。与XML紧密相关的是扩展样式表语言(XSL),其文档转换组件(XSLT)具有强大的功能,能够对文档进行编码转换以便进行安全传输。本文通过在XML中的实际应用来加以验证;同时本文提出了一种将XSLT作为插件考虑的文档传输体系;并且标识出了为适应这种应用需求,要用到的XSLT的相关属性。通过利用XSLT所具有的扩展函数,可知XSLT是最适合XML数据安全应用的方案。

关键词 扩展函数,XML,XSLT,Security

Research and Application of XML Security Using XSLT

CHEN Ning GE Jun-Wei DENG Ning-Bo

(College of Computer, Chongqing University of Posts &Telecommunications, Chongqing 400065)

Abstract As a result of the eXtensible Markup Language (XML)occurred, it is possible to enable transfer of data amongst heterogeneous environments. During application, one important requirement is for the security of documents in transit. Closely associated with XML is the eXtensible Stylesheet Language (XSL), whose document transformation component (XSLT)may well have sufficient functionality to perform all reasonable cryptographic transformations to deliver a desired level document security. The paper examines this question by describing a real world XML application; proposing a document transfer architecture into which XSLT can be plugged-in; and identifying those features of XSLT which must be applied to meet the application requirements. At last the author conclude that XSLT is only just adequate in the proposed scenario; and then only by making use of its "extension functions" capability.

Keywords Extension functions, XML, XSLT, Security

1 引言

XML已经问世几年了,它提供了一个经由网络来提升信息传输性能的体系结构,这种源于W3C的标准,允许用户个性化地组织信息。其结构化信息的实例就是XML文档。一个XML文档必须符合特定的语法和语义,这些规范在W3C第6次建议的XML1.0(第二版)中有明确规定[1]。由于支持系统无关的字符集(Unicode),使得XML文档在阅览时将比二进制文件更具有亲和力,但同时它也带来了信息隐私方面的问题。至少,一些敏感信息不应当以人所能见的方式出现。要达到这种目的,一种简单而有效的方法就是对XML数据元素进行加密。元素加密应在各自独立的基础上进行。在一个XML文档中,数据元素可以放在两个地方:

一是作为标记(tags)属性;二是作为内容元素的文本。

我们需要一个安全的XML文档,并且其结构与原始的XML文档要一致。通过加密,能从任何安全措施中分离出相关应用。其原理是以XML应用的实践经验为基础,例如数据套接字(Data Sockets)方案[7],在该方案中文档由来源于不同数据源的元素组成,通过一个不与那些元素的数据内容相关联的中介,在文档的个性化元素的可见性方面,提供了一种容易获得的访问控制机制。

XSLT(可扩展样式表语言转换)是W3C的一个文档操作语言规范,能够重构文档和在其元素上执行运算[6]。XSLT是否能将一个XML文档表示成加密的形式或由一个加密的文档还原出原始的XML文档,并在XSLT的框架中将文档处理的各个方面统一起来呢?这导致了本文将要讨论的问题:使用XSLT在XML文档与安全的格式之间进行相互转换包含哪些问题?XSLT适合解决这些问题吗?

如果我们仅仅关心加密或解密成另一XML文

陈 宁 硕士研究生,研究方向为网络安全、网络管理;葛君伟 教授,博士,硕士生导师,研究方向为TMN、分布式对象技术、网络安全;邓宁波 硕士研究生,研究方向为网络管理、网络安全。

档的转换操作，那么 XSLT 的优势很明显，它能提供诸如表示格式，还有文档转换的各个方面的东西。特别是它能避免使用专用软件，这些软件需要专门配置和维护：一旦拥有了一个标准的 XSLT 处理机，安全操作规范就可以限制成样式表的形式。

本文中考虑了一个电子商务的范例，在该例中，中介或代理通过对输入的 XML 文档进行解析，生成一些较小的 XML 文档（构成一个事务），或相应地，中介将这些解析后的 XML 文档组合起来。出于尊重隐私的考虑，XML 文档中包含的数据对于中介而言应该是不可见的，但文档结构，包括个性化元素的标识要可见，以使其正常操作文档。在该范例中，对样式表应用加密或解密是可行的，以使信息如设想的那样，对于中介而言是透明的，同时将信息的控制权交给其所有者或来源地：对样式表的加解密应用使得访问控制机制成为可能；这种过程可以通过不同的密钥，甚至加密程序，来实现对元素进行逐个加密。下面本文将在满足上述要求的情况下，通过执行安全转换来验证 XSLT 的能力，同时也指出它的局限性。

2 相关工作

虽然这项研究确认了使用 XSLT 来获得 XML 文档安全的适用性，但它并不是唯一获得 XML 文档安全的方式。还有一个简单的方法，能在几乎任何一种编程语言中实现，它是以元素方式加密为基础，并且运用了 XML 处理机。这种方案很明显的好处就是在对文档进行构建和处理时减少了复杂度。不足之处在于这种方案要求在系统的每个点上都要有这样的包（XML 文档安全被处理的地方）并且需要一种新的机制，该机制是为了在一个源 XML 文档中确定加密选择。

这种类型的一个执行例子在 W3C 中能够找到，XML 加密，将涉及到的安全包含到 XML 处理机中，在一个 XML 文档中被其属性所控制（为了初始化和引用的目的）[5]。IBM 公司的 XML 安全套件就是这种处理机产品[3]。

2.1 元素方式加密

元素方式的加密是指运用加密算法，经由一个 XML 文档生成安全 XML 文档的过程[4]。这个概念最早被 Maruyama 和 Imamura 提出，在这种方案中对信息的有效性有更严格的要求，并且结果文档与源文档相比允许拥有不同的结构。在 XML 文档段中，应用元素方式加密与执行语义操作相似。

在 Maruyama 和 Imamura 提出的元素方式加密的方案中，信息的有效性（通过文档类型定义 DTD 的使用）被提出，作为一种在 XML 文档中确认内含信息的方法，有其存在价值。但他们却未考虑由此衍生的 XML 文档的有效性问题。

3 加密转换

这部分将验证纯 XSLT 在加密转换中的便捷应用，提出的适合策略是运用扩展函数，该函数允许要求运算的 XSLT 代码能以另一种语言表示。针对此问题提出的模型如图1所示：一个 XML 源文档在进行转换（该转换包含在样式表中）后，生成了一个加密的 XML 文档（即安全的 XML 文档）；接下来样式表把一个加密的 XML 文档再转换回原始形式。

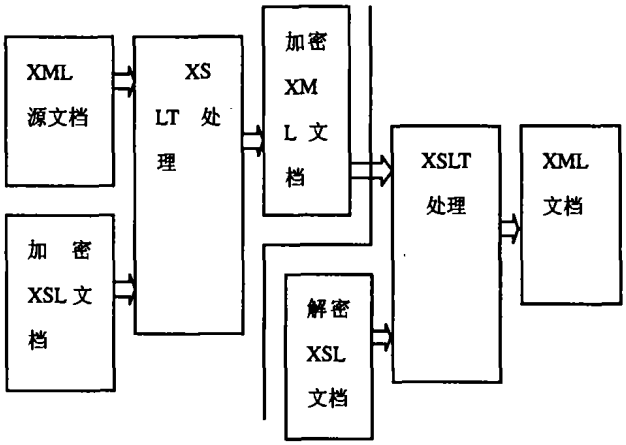

图1 XML 加密和解密

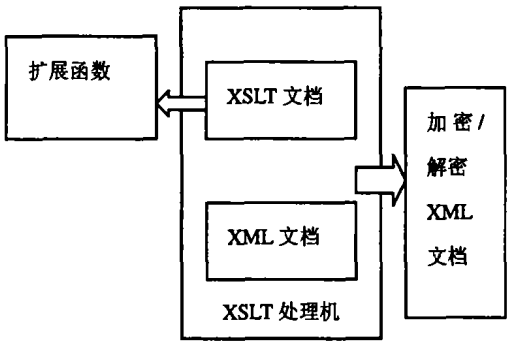

图2 在样式表框架中的扩展函数

完全符合 XML 语法的加解密样式表可以构造出来，但实际上效率不高，因对没有指派声明的加密算法进行编程，其本身很复杂，甚至在样式表中一个简单的程序就要明显增加复杂度，我们只需考虑使用 XSL 样式表进行 N! 的运算，就能确信这点。

3.1 扩展函数

为了在不影响样式表复杂度的前提下提供便捷的编程语言功能，扩展函数被利用起来[2]。显然，要完成这样的功能，需要有两种不同的扩展函数：

第一种函数用于在密码库和样式表之间作为加密接口；第2种函数用于在密码库和样式表之间作为解密接口；

扩展函数的运用与元素方式的加密类似。样式表对 XML 文档实施加解密的过程有以下几步：

1 在相应的 XML 文档中，进行与所有标记相

匹配的模板声明。这一步必须要有，以便保留在原始XML文档中的内含结构。右图的示例是一个XML文档：

```
〈?xml
version-"1.0"?〉
〈tag1〉
    〈tag2/〉
```

其中包含了两种不同的标记；tag1和tag2，为了在输出文档中被重新生成，必须使每个标记与一个模板相匹配。否则XSLT处理机将丢弃那些未明确匹配的标记。这种情况最不希望发生。

2 扩展函数针对文本字符串实施所有的加解密工作。

3 属性标识和重新生成。这点与第一点很相似，在一个XML源文档中包含的属性也必须在输出文档中被重新生成，否则将被丢弃(这就会造成信息丢失)。

图2标出了在样式表框架中的扩展函数，XSLT处理机接收到一个XSLT文档(一个样式表)和一个XML文档后，运用扩展函数来生成一个加密或解密的XML文档。

显而易见，这种XSLT解决方案的思路与在第2部分描述的用户内建的加密套件的思路很相似；比如，像解析然后加密或解密的过程。但这种方法也减少了可能已从纯粹的XSLT解决方案中获得的优势，在XSLT处理机中的预处理逻辑仍然以一种合理的透明的方式，连接含有XML解析接口的加密函数，比如SAX[10]。因此，进一步的工作还有待去完成，正如在第2部分中所指出的一样。

4 元素方式加密

在这一部分，主要讨论构建保护加密的需求是怎样遇到的，并且检验在经由XSLT处理机运作的转换中有哪些受限的地方。

如果我们要求一个经过加密的XML文档与其解密形式有一样的XML结构，不论其是否被加密，都能被文档处理程序操作。那么在这样的文档中包含的值就要从文档结构中剥离出来。比如，在范例中基于标识文档元素值的XML文档被接收后转发，所有的值(除了用来标识元素的外)都是不相关的(对于前转任务而言)。在加密文档和未加密文档之间的区别在于，在加密文档中，需要的原始数据块难以破译。这些数据块可以被标示成属性或作为内容元素的文本节点内容。例如：

〈tag attribute="data"〉 data 〈/tag〉

此处的data表示能够被加密的数据。该数据存在于两个地方，一个是作为元素的内容，另一个是作为attribute属性的值。

与大量的加密数据块比较起来，人们通常认为少量的加密数据很容易泄密。在这项研究中，这个问题不用担心，因为许多安全套件弥补了对少量数据进行加密所带来的风险(比如，在加密前将数据填充至特定长度)。

这种类型的加密应用在各加密数据块之间可以是相互独立的。例如，在一个XML文档中的每种属性可能应用了不同的加密算法和密钥。此外，数据块也不一定非得加密，这还要依赖于信息源的要求。

在该验证过程中，元素方式的加密概念还需要精炼。起初，元素方式的加密被设想为对一个XML段进行加密(包括该节点本身和其所有子节点)。但这仍然未解决这样一个问题：在样式表被用于一个XML文档之前，XML处理机必须解析此XML文档，生成一个DOM(文档对象模型)对象的表示[8]。在应用样式表时，一个XSLT处理机通过DOM对象逐步应用相应模板，如同标记需要匹配一样(在XML文档中的标记与样式表内的模板要相匹配)。这意味着，在样式表被应用之前，一个XML文档就不再以一个XML文档的形式存在。同时经由一个DOM对象的段能够实现加密函数，但这也会产生新问题：一个DOM对象的段(子树)并不是一个文本元素；当尽力恢复片段时，额外的处理不能实施(为了将一个子树还原成原始的XML格式)。

初步实验中，DOM段从XML文档中被样式表抽取出来，然后被加密。该加密的DOM段被直接解释。然而，在连续对DOM对象(存在于一个加密的XML文档中)加密时，关于对象的信息将会丢失，并且尝试去恢复DOM段时，不会成功。

未使用附加的扩展函数并不意味着发现了在此DOM段上执行额外的评估(比如，值抽取)。

由于具有额外的复杂度，元素方式的加密不适用于结构化的元素中，例如文档的非平凡子树。

5 运行测试环境

在本文中讨论的XSLT各个方面，都通过对大量示范样式表的指定和实施进行了测试或验证。这些测试中使用的XSLT处理机是jd.xslt 1.2.1版本。jd.xslt支持XSLT1.1并且内置于java编程语言中[2]。

正如在3.1中所描述的，与提供附加功能外挂于样式表的XSLT比较，扩展函数被编码写进编程语言。XSLT扩展函数是处理机的执行细节。在使扩展函数应用标准化的过程中，XSLT1.1为了声明扩展函数，引入了〈xsl:script〉标记。为使XSLT处理机能够调用以特定语言写的扩展函数，处理机必须绑定该语言的内置扩展函数。语言绑定是可选的，而且保留了处理机细节。

在本范例中使用的扩展函数是用java写的。这

样做是因为 jd.xslt(XSLT 处理机)是在 java 中构建并且提供了针对该语言的绑定。在该范例中,由于 jd.xslt 的运用要求使用 JRE,java 扩展函数就使用了与 jd.xslt 一样的 java 运行环境(JRE)。

6 测试过程

```
〈?xml version="1.0"?〉
〈document〉
        〈customer id="1"〉
            〈surName〉Bloggs〈/surName〉
            〈firstName〉Joe〈/firstName〉
        〈/customer〉
        〈customer id="2"〉
            〈surName〉Doe〈/surName〉
            〈firstName〉John〈/firstName〉
        〈/customer〉
〈/document〉
```

图3

```
〈xsl:script implements-perfix="security"
language="java"src="java:test"/〉
…
〈xsl:template match="surName"〉
    〈xsl:text
    disable-output-escaping="yes"
    〉&lt;surName&gt;〈/xsl:text〉
    〈xsl:value-of select="security:encrypt(text())"/〉
    〈xsl: ext disable-output-escaping =" yes"〉 &lt;/
    surName&gt;
    〈/xsl:text〉
〈/xsl:template〉
```

图4

这部分给出了测试中的每个步骤。包含了 java 扩展函数代码的文件首先使用 javac 编译器(在 Java Software Development Kit,SDK 中)进行编译。

XML 源文档和加密样式表作为 jd.xslt 的输入。从扩展函数创建的类文件原代码一旦被编译,就必须被包含在处理机的类路径上。通过对 XML 源文档运用加密样式表,就生成了一个安全的 XML 文档,同时还生成一个"keystore"文件,用来存放加密的密钥。

安全的 XML 文档和加密样式表又输入到 jd.xslt 中,包含有扩展函数的类文件必须被包含在 XSLT 处理机的类路径上。通过转换,原始的 XML 文档被重新生成。XML 源文档如图3。在测试中使用的加解密样式表很相似。两者不同之处在于使用了不同的扩展函数(例如,加密样式表使用加密函数,而解密样式表使用解密函数)。上述代码描述了部分 XML 文档,我们将使用其来说明一个典型的测试用例;元素 surName 将被加密。样式表加密抽取代码如图4。

以上代码描述了出现在相应样式表段内的必要转换。加密函数 encrypt()作为扩展函数被标示,利用命名空间的前缀 security,该前缀自己被 xsl:script 元素所定义并且链接到相关的被编译过的类。

结束语 当 XSLT 拥有操作 XML 文档的结构和内容的能力时,应尽力抽取出额外的功能,比如导致了难以忍受的复杂度并且限制了文档加密方式的加密算法的运用。

扩展函数的运用使前一问题得到解决,通过提供扩充 XSLT 性能的组件。但这个方案一定程度上增加了技术难度,目前,在 XSLT 规范中扩展函数还需要被明确地定义。作为一项技术,XSLT 还有待完善的地方。

参 考 文 献

1 W3C. W3C Recommendation 6 October2000. http://www.w3.org/TR/2000/REC-xml

2 W3C. XSL Transformations (XSLT)Version 1.1. http://www.w3.org/TR/2001/WD-xslt-20010824/

3 Tidwell D. The XML Security Suite:Increasing the security of e-business. http://www-4.ibm.com/software/ developer/library/xmlsecuritysuite/index.html,April 2000

4 Maruyama H, Imamura T. Element-Wise XML Encryption. http://lists.w3.org/Archives/Public/xmlencryption/2000Apr/att-0005/01-xmlenc.html

5 W3C. XML Encryption Syntax and Processing. WGWorking Draft,June 2001

6 White C 著,王健,王军 译. XSLT 从入门到精通.电子工业出版社,2003

7 Bryan G,Curry J,McGregor C,Holdsworth D,Sharply R. Using XML to facilitate information management across multiple local government agencies. Hawaii International Conference on System Sciences,Hawaii,United States,Jan. 2002.1544~1553

8 W3C. Document Object Model (DOM)Level 3 Core Specification Version 1.0. W3C Working Draft,June 2001

9 W3C. XSL Transformations (XSLT)Version 2.0. W3C Working Draft,April 2002

10 Brownell D. SAX. http://saxproject.org,Jan. 2002