第 33 卷 第 11 期 计 算 机 工 程 2007 年 6 月
Vol.33 No.11 Computer Engineering June 2007
·安全技术· 文章编号：1000—3428(2007)11—0154—03 文献标识码：A 中图分类号：TP393

基于 XML 数字签名的应用模型设计与分析

陈文兵
(南京信息工程大学数理学院，南京 210044)

摘 要：介绍了利用 XML 数字签名设计带有安全要求的应用系统的方法。分析了推荐的 XML 签名和验证模型在对 XML 消息进行签名和验证时存在的不足。给出了 SOAP 数字签名流验证的 CXML/SSoap_SV 系统的设计和实现思想。根据实验结果分析得出结论，流验证模型提高了应用系统的性能和系统的存储效率。

关键词：XML 规范化；XML 数字签名；XML 加密

Design and Analysis of Application Model Based on XML Digital Signature

CHEN Wenbing
(College of Mathematics and Physics, Nanjing University of Information Science & Technology, Nanjing 210044)

【Abstract】 This paper introduces a type of method using XML digital signature to design application system with security request. It analyses the defect using the recommended XML digital signature & validation model to sign and validate XML message. It presents the ideal of design and implement for CXML/SSoap_SV system based SOAP digital signature & stream validation. It concludes that the streaming validation model can enhance the performance of application system, and increase the memory efficiency of the system.

【Key words】 XML canonicalization; XML digital signature; XML encrypt

安全是 Web Services(WS)和 Grid Services(GS)的一个至关重要的特性。为了支持消息层、端到端的安全，SOAP 数字签名和 WS 安全定义了利用 XML 安全语法进行签名或加密 SOAP 消息的机制。然而，当前机制的实现在应用性能上比较低，有人做过这方面的实验证明消息层安全确实会降低基于 GS 的 SOAP 性能，故在发送长签名的 XML 消息时消息层安全的实际应用几乎无法成为现实。

本文提出一个新的 SOAP 数字签名验证流处理模型。该模型由流规范化和优化的 SOAP 签名验证组成。本文的流规范化在 XML 解析期间不需要建立完整的 DOM 就能够以流形式规范 SOAP 消息。文中的方法适用于大部分 SOAP 消息的处理。

1 目标

1.1 XML 签名简介

XML 签名[1,2]是将一个数字签名结果同任意(通常是 XML)数据进行关联的数字签名。其语法定义结构如下，结构中各标记的含义可参见文献[1,2]。

```
<Signature ID?>
  <SignedInfo>
    <CanonicalizationMethod/>
    <SignatureMethod/>
    (<Reference URI? >
      (<Transforms/>)?
      <DigestMethod/>
      <DigestValue/>
    </Reference>)+
  </SignedInfo>
  <SignatureValue/>
  (<KeyInfo/>)?
  (<Object ID? />)*
</Signature>
```

一个 XML 签名核心验证过程必须完成：(1)引用验证，指对包含在<SignedInfo>元素内每个<Reference>中摘要的验证；(2)签名验证，指对包含在<SignatureValue>元素内加密签名的验证。具体的引用验证和签名验证步骤参见文献[1,2]。

由上述算法可知，在签名验证中包括两个主要数据模型：XPath 节点集和字节流。二者可以相互转化，如图 1 所示。

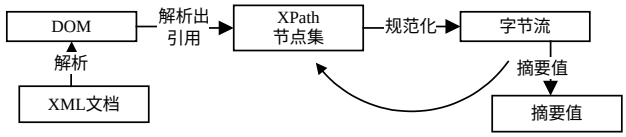

图 1 推荐的 XML 签名数据流处理模型

推荐的 XPath 节点集很容易表示出一个 XML 文档中元素的子集，直接实现方法是用 DOM 节点描述该节点集。故签名验证可能要求一个变换序列，该序列含有在 DOM 和字节流之间的多重相互变换。若采用 DOM 回存 XPath 节点集，则验证算法将降低应用性能。

1.2 SOAP 数字签名及 WS 安全

SOAP数字签名[3]利用XML签名语法签名SOAP1.1 消 息[4]。SOAP头实体<Signature>定义一个数字签名，它符合XML-签名规范。通用SOAP数字签名消息结构如下所示，说明了签

基金项目：南京信息工程大学科研基金资助项目(QD31)
作者简介：陈文兵(1964－)，男，副教授、硕士，主研方向：信息处理与数值计算
收稿日期：2006-07-30 **E-mail：**chenwb6403@hotmail.com

名SOAP消息的通用消息模板，该模板中SOAP封装的<Head>构成了签名值并引用<Body>中的数据对象。

```
<Envelope>
    <Head>
    <Signature>
        <SignedInfo>
            <Reference URL=”#doc” />
        </SignedInfo>
        <SignatureValue>
        …
        </Signaturevalue>
        <KeyInfo />
    </Signature>
    </Head>
    <Body Id=”doc”>
    …
    </Body>
</Envelope>
```

上面的结构称为流模板，模板中 SOAP 数字签名定义了两个约束条件：<Reference>元素要先于数据的定义；在<Transforms>中只允许流变换。

流变换定义为一个变换算法可以通过部分输入即可正确地生成部分输出。规范化是可流的且通过流XPath查询技术XPath过滤也可转化为可流的[5]。

1.3 规范化

XML 1.0 建议及其中的名字空间[4]定义了表示同一信息的多种语法构造方法，如在元素的开始标记中属性的次序不影响元素的实际内容。意味着同一个XML文档可得到不同的字节流表示但其实质内容完全一样，签名计算却由于这种表达形式上的差异会导致结果的不同。XML规范化[6](简记为C14N)主要用来解决这种形式上的差异。C14N定义一个规则集用来将一个XML文档或一个文档子集转换成规范形并保证相同内容的文档具有唯一的规范形。故在应用中处理的是文档的规范形而不是原始文档。该规则集参考文献[6]。

1.4 流化的目的及具体处理方法

如前所述，推荐验证模型验证 XML 签名可能会导致费时费内存的 XML 处理和数据变换。而大多数 GS 应用不需要变换，只要一个简单的 XML 签名处理。但即便如此，由通用 SOAP 数字签名消息结构可以看出推荐验证模型需要解析 XML 文档、构建 DOM，再由 DOM 创建 XPath 节点集并将其规范化成字节流等过程。因此，在内存中至少要建立一个 DOM 树、一个节点集，以及两个规范形并驻留内存。当消息变大时，所需内存量就会成为性能问题。若再引入变换就会使得这一问题进一步恶化。这足以表明，与使用传输层的安全实现相比，推荐模型实现 XML 签名将会使简单 GS 调用的响应时间性能降低，其中主要时间开销在规范化上。但一个流验证模型，特别是流规范化，在处理期间不用构建上述的数据结构。此外，大多数签名 SOAP 消息符合流模板规范，本文基于流规范化给出了一个有效的流模板签名验证模型。

2 架构

本文系统中采用一个流处理模型进行规范化。与基于一个完整 DOM 实现相比，一个流模型一旦有了所需的信息就可以立即生成部分规范的 XML。将会看到生成的 XML 令牌(即开始标记)的规范形仅需要有限的语义和上下文信息，而这些上下文信息可以由 XML 解析的上下文显式或隐式地获得，因此在 XML 解析期间可以利用部分上下文进行流规范化。为了得到较高的性能，本文的 XML 解析器(CXML)中嵌入了规范化功能以使规范化处理同解析器共享解析上下文并在 XML 解析期间并发地运行，XML 规范形片断在 XML 解析期间同时得到。CXML 提供了一个类似于 SAX 的接口，但后者没有流规范化功能。该应用将会得到一个事件序列以及流规范化的结果(即规范数据或它的摘要)。

流 SOAP 签名验证器(SSoap_SV)是具体的流化模板签名验证器，实际应用中它位于 CXML 和应用程序之间，扮演一个中间节点角色，位于 CXML 发出的事件响应链之间。SSoap_SV 检验那些与事件相关的 SOAP 签名语法并验证签名。与此同时不管签名验证是否完成，SSoap_SV 立即向应用传送已经收到的每个事件。因此，在签名还未验证时，应用程序就会收到一系列属于签名范畴的事件。对那些 SSoap_SV 将其标记为未验证的事件，应用程序收到该事件时其响应由于事先已经知道事件的类型，因此可以避免做某些不可恢复的动作。图 2 说明了上述流处理模型的基本架构和数据流程。

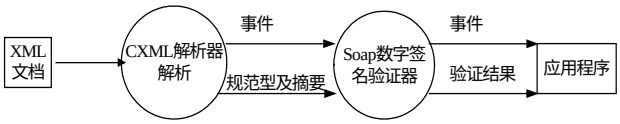

图 2 流验证模型的架构

3 实现

3.1 CXML 解析器

XML 语法中所用的标记符可以使人很容易地通过递归的方法构造它的解析器，在解析器中将每个过程同语法的每个非结束符关联并以自顶向下的语法分析方法递归地执行该过程集处理其输入。在 CXML 中对 XML 1.0 语法文法的每个非结束符，首先要基于它们的生成规则建立转换图。例如，对元素 Element、空元素 EmptyElemTag、开始标记 Stag，按如下规则生成：

Element::= EmptyElemTag | STag 内容 ETag

EmptyElemTag::=’<’ Name(S Attribute)* S?’/>’

STag ::= ’<’ Name (S Attribute)* S? ’>’

经适当的分解后其文法转换图如图 3 所示。

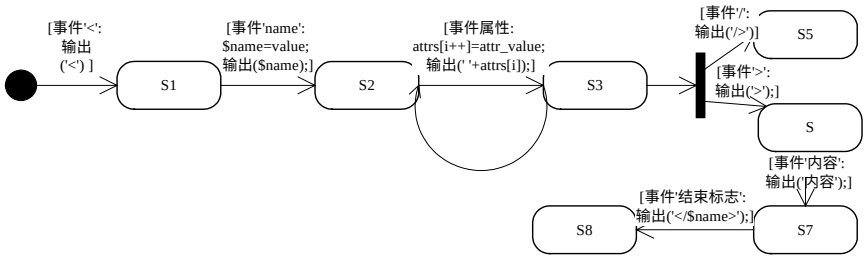

图 3 转换图

根据图 3，对非终点的递归过程可按下述规则进行构造：

(1)对终点上的转换意味着，如果该终点是接下来的输入标志应再调用该转换。

(2)对一个非终点 A 上的转换是对 A 过程调用。

为了支持 XML 规范中的名字空间，CXML 必须维护一个名字空间定义栈来记录解析期间名字空间声明。与 SAX 类似，对元素的每个开(闭)标记，CXML 生成相应的开始(结束)事件。元素的开始事件与(属性名，属性值)数对列表一起产生。对由开闭标记封装的正文内容，该解析器生成一个正文解析事件。

3.2 流规范化处理

根据 C14N 规范化规则，可以将规则分成两类：基于语法的和基于上下文的。

基于语法的规则，意味着转换所需的只是当前局部语义上下文，例如转换图中的当前状态。断行标准化、属性值标准化以及空元素转换等。而基于上下文的规则需要解析上下文，如名字空间上下文及 DTD 类型上下文。属性的字典次序要求名字空间前缀对应的 URI，且增加的缺省属性值要求 DTD 定义。根据 XML 语法，在执行基于上下文的规则时，名字空间上下文和 DTD 类型上下文都可用。在 CXML 中，转换图的当前状态隐式地为所有基于语法的规范化规则提供了所需的信息，而由 CXML 维护的名字空间栈和 DTD 解析结果为基于上下文的规范化规则提供了显式的信息。这些性质为执行流处理提供了充要条件。

为在 XML 解析器内嵌入规范化处理功能，将转换图中的转换与解析期间触发的相关动作进行规范化关联，被关联的规范化动作随同状态转换附带执行。动作的输出(语法表示的规范形)将存入缓冲。将图 3 转换图扩展成图 4 以便能正确地规范化 XML 文档中的开始标记。

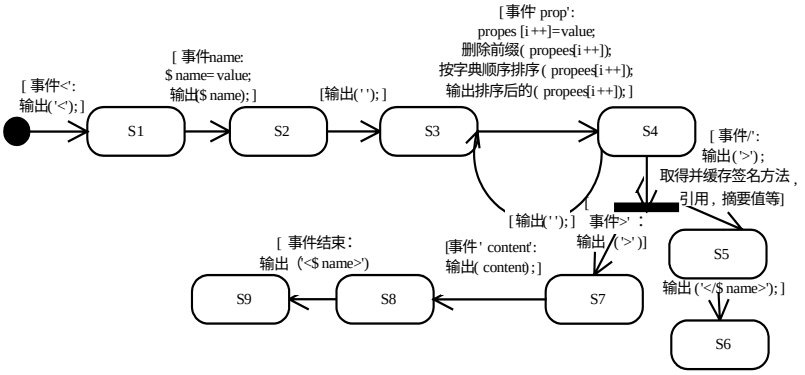

图 4 被规范化处理动作扩展了的转换图

被关联的动作解释如下：

(1)规范化要求，在一个开始标记中的所有名字空间的声明要放在(以文档次序)所有其他属性的前面。因此，在状态 4，当它确定一个属性是不是一个名字空间声明时，被解析的值既可以被放进名字空间缓冲也可以放入属性缓冲中。

(2)规范化要求删除多余的名字空间声明并在名字空间声明和属性上强制执行字典顺序。故在状态 4，当在开始标记中的所有属性和名字空间声明都已被解析并分别缓存时，在开始标记中出现的所有前缀都能够被解析并能够检查出多余的名字空间声明且删除这些前缀。接着排序并以字典顺序分别输出名字空间声明和对应的属性。

(3)规范化要求空元素被转换成开始结束标记对。因此当达到状态 6 时，将输出附加的“ </$name> ”。

因此，解析之后 XML 规范形片断可以同时得到，可见流模型不需要构建费时的数据结构。

3.3 C14N 会话

在 C14N 会话期间，解析器将为输入的 XML 标记生成规范形。CXML 提供了一个接口允许应用决定何时打开或关闭 C14N 会话。当会话打开其后的 XML 片断将会得到解析并自动进行规范化直至该会话被关闭且规范化结果被存入缓冲。应用程序可将会话选项设为 DIGEST-ONLY 以指定该应用程序仅需从规范化中得到摘要。对基于 XML 安全的应用程序而言，消息中的摘要才是其真正感兴趣的内容。在应用程序中 CXML 保持固定大小的块缓冲存储规范化的输出。当缓冲满时应用程序就会调用消息摘要算法(SHA1)增量计算该摘要，而块缓冲则可为后续的规范化重复利用。增量摘要算法保持系统的空间复杂性不变，与 XML 消息大小无关。

C14N 流规范化机制灵活，如果应用程序在文档开始打开 C14N 会话而在文档结尾关闭会话就会得到整个文档的规范形。无论何时，遇到一个开标记(如<books>)应用就会打开 C14N 会话,而遇到闭标记</books>就关闭会话，若 XPath 表达式“ /books ”不是递归结构，则其结果就是其节点集的规范形。

3.4 流 SOAP 签名验证

SOAP 消息符合流化模板，该模板为流签名验证提供了保证，由于在以文档次序签名的数据前面会遇到所有的元数据，因此特别设计 SSoap_SV 来处理流化模板。SSoap_SV 是一个自动构件，运行时识别 Soap 数字签名语法并验证签名。如 CXML 解析器一样，在构件中也将转换与动作进行关联，只是这里将面向 XML 语法的动作换成了面向 SOAP 签名语法的动作。构件的结构如图 5 所示。下面按处理顺序解释其关键转换：

(1)状态 1：当遇到<SignedInfo>时，由于需要一个规范的<SignedInfo> </SignedInfo>片断，因此打开 C14N 会话。在片断上应用加密的签名算法。

(2)从状态 2 到状态 3：签名和摘要计算的元数据(如 CanonicalizationMethod 或 Signature Method)将存入临时变量用于后续的计算。特别地，当遇到<Reference>时，保存属性“ URI ”的值以供将来定位后续流中签名数据对象。

(3)在状态 3：当遇到</SignedInfo>， 关闭 C14N 会话。而规范的<SignedInfo>... </SignedInfo>片断结果可以从解析器中检索到。

(4)在状态 6：当遇到</Signature>时，调用签名生成算法(即 RSASHA1)以在元素<SignedInfo>的规范形上计算签名并将其结果同从<SiganureValue>中检索到的签名值进行对比。

(5)在状态 7：当遇到具有匹配 ID 属性的元素时，将再次设置新的 C14N 会话而此时由于仅需要部分规范形的摘要，因此将设置 DIGEST-ONLY 选项。

(6)新的会话一直保持打开直到遇到对应的结束标记。接着可以从解析器中检索摘要并将它与从<Digest>中得到的供引用验证的摘要进行对比。

运行自动构件之后需要校验：SOAP 消息中数字签名语法的正确性；消息的签名验证和引用验证。

这里假定了在签名模板中没有变换，实际上在上面的模型中仅解析出了签名数据。但如前所述 SSoap_SV 可用流 XPath 查询进行扩展以支持大多数流方式 XPath 过滤变换。SSoap_SV 已经生成了一个隐式的节点集，对签名数据对象其回存是 SAX 事件序列而不是 DOM 且以流方式使用 C14N 会话。限于篇幅系统的性能分析不再讨论。 （下转第 162 页）