XML 签名在电子病历系统安全中的应用①

XML Signature for Security In Electronic Medical Record Information System

郑　重　薛万国　（解放军总医院计算机室 100853）

摘要：XML 签名定义了对 XML 文档签名语法和处理方法，能够保证 XML 电子病历文档的完整性和不可否认性。本文讨论了电子病历中 XML 签名及验证的过程、方法及结果。

关键词：电子病历 XML 签名和验证

1　引言

电子病历（Electronic Medical Record）是以电子化方式管理的有关个人终生健康状态和医疗保健行为的信息，它可在医疗中作为主要的信息源取代纸张病历，提供超越纸张病历的服务，满足所有的医疗、法律和管理需求。电子病历系统是一种信息系统，它主要对医院信息系统中的电子病历信息进行采集、归档，对归档后的病历库进行管理并提供浏览查找等相关服务。

“如何保证病历记录不被他人篡改，如何保证医师对病历的修改不可抵赖”，也就是病历的完整性和不可否认性是电子病历系统安全中两个重要的问题，数字签名提供了很好的解决方案。

数字签名是一种密码技术，它利用一个密钥对给一段信息（常常是经过散列后的 Hash 值）进行加解密。签名者唯一拥有私钥，利用它进行加密，验证者只需取得签名者的公钥证书解密后对比签名前信息就可以验证签名者身份的合法性。本电子病历系统中电子病历文档是以 XML 文件方式保存的，将电子病历文档以 XML 方式保存有几点好处：XML 作为结构化描述语言能够保留病历的信息结构；XML 对病历的表达更直接、直观；XML 结构有利于病历的院际交换和适应病历内容的变化和发展。对 XML 文档的数字签名国际上有一个推荐标准——“XML – Signature Syntax and Processing（XML Signature）”，是由负责 Web 技术标准制订的 World Wide Web Consortium（W3C）组织于 2002 年 2 月 14 号发布的。这个标准提供了 XML 文档的签名语法，使得文档在签名前后都是结构化的。本文按照 W3C 标准用 java 语言实现了对 XML 电子病历的数字签名。

2　签名过程

2.1　病历 XML 文档的说明

本系统中单个病人在院期间的医疗记录以一个 XML 文档的形式在出院时保存下来。XML 文档由一个个医疗文档组成，每一个医疗文档对应着一种医疗行为，比如入院记录、检查报告等等。文档结构如下：<cdocs>为根节点，代表病人的整个医疗文档，<cdoc>作为<cdocs>的子节点代表一个个医疗文档。每个医疗文档也就是<cdoc>节点内容对应一个签名的基本内容单位。其 XML 文档 DTD 文件如下所示（部分）：

```
<! ELEMENT cdocs (cdoc) + >
<! ATTLIST cdocs
    pid CDATA #REQUIRED
    vid CDATA #REQUIRED
    ……
>
<! ELEMENT cdoc (pidinf, (demog | inhabst | ords | vtsign | labrpt | exrpt | admr | progres)) >
```

医疗文档节点（病人简要信息（病人自然信息|住院病案首页|医嘱记录单|生命体征记录|检验报告|检查报告|入院记录|病程记录））

以下提供的是包含一个“出院记录”医疗文档的

① 本项目得到北京市自然科学基金（项目编号 4012012）和军队科研基金（项目编号 01G007 – 1）资助

一段电子病历文档实例：

```
<? xml version = " 1. 0" encoding = " GB2312" ? >
<! DOCTYPE cdocs SYSTEM " mr. dtd" >
<? xml - stylesheet type = " text/xsl" href = " MR. XSL" ? >
<cdocs pid = " 319761" vid = " 1" status = " editing" cdt = " 2002 - 02 - 11" udt = " 2003 - 02 - 12" adt = " 2003 - 02 - 13" secley = " 1" >
<cdoc docid = " cdoc0004" ver = " 1" type = " 病程记录" status = " edit" title = " 出院记录" cdt = " 2003 - 04 - 28" udt = " 2003 - 04 - 28" fdt = " 2002 - 04 - 28" creator = " taos" >
<pidinf >
<pid >319761 </pid >
<nm > 张三 </nm >
<sex > 男性 </sex >
<dob >1954 年 10 月 7 日 </dob >
</pidinf >
<progres type = " dchsumm" subject = " 出院记录" >
<dchsumm >
< extref type = " pic" ref = " pic319761_0001" height = " 194" width = " 229" / >
<paragraph > 张三，男性，34 岁，汉族，已婚，内蒙古籍。本院脑外科医生。因右手外伤术后 1 小时。2003 年 4 月 24 日入院，2003 - 4 - 28 日出院，共住院 5 天。 </paragraph >
<paragraph > 入院时情况：一般情况好 ，心肺检查未见异常，腹部平坦，肝脾肋下未及。
右手小鱼际有一长 5cm、宽 3 cm 的逆行撕脱皮肤，有活动性出血，右手各指无麻木感，痛觉存在，无感觉缺失，手指活动好。桡动脉搏动良好，手指末梢充盈好。 </paragraph >
<paragraph > 入院诊断：右手皮肤撕脱伤术后 </paragraph >
<paragraph > 诊疗过程：入院经抗感染治疗，伤口无感染，皮肤无坏死。住院期间未发生院内感染及并发症。 </paragraph >
<paragraph > 出院时情况：一般情况好，生命体征平稳，手术切口换药，无渗出，皮肤切口对合佳，皮肤感觉手指运动正常。 </paragraph >
<paragraph > 出院诊断：右手皮肤撕脱伤术后 </paragraph >
<paragraph > 出院后注意事项：继续口服抗凝药物，术后两周拆线，功能锻炼，门诊随诊。 </paragraph >
</dchsumm >
</progres >
</cdoc >
.....
</cdocs >
```

2.2 密钥的获得

密钥是签名必不可少的元素。对于密钥的存放有几种方式，JDK 支持两种存放方式：pfx 方式是指以 pkcs#12 格式存储的证书和相应私钥；jks 方式是 java 密钥库 keystore 方式存放证书和相应私钥。本文以 jks 方式为例，其中 storePath 参数表示证书文件路径，storePass 为密钥库 keystore 的密码，KeyPass 为签名证书的密码，alias 为证书别名。以下是获取私钥的一段源码：

```
public static Key getKey ( String storePath, String storePass, String KeyPass, String alias) throws
KeyStoreException, CertificateException, UnrecoverableKeyException, FileNotFoundException,
NoSuchAlgorithmException, IOException{
KeyStore ks = KeyStore. getInstance( " JKS" ) ;
ks. load( new FileInputStream( storePath) , storePass. toCharArray( ) ) ;
Key signkey = ks. getKey ( alias, KeyPass. toCharArray( ) ) ;
if ( signkey = = null) {
System. err. println( " Could not get a key: " + alias) ;
System. exit( 1) ;
}
return signkey;
}
```

2.3 签名类型的讨论

W3C 的 XML 签名标准定义了三种签名类型：封装的 xml 签名（Enveloped Signature），封装用 xml 签名（Enveloping Signature），分离的 xml 签名（Detached Signature）（参考 RFC3275）：

(1) 封装的 xml 签名。签名元素在待签名的 xml 文档中,具有以下的 xml 格式:

```
<original_document>
    <Signature>…….</Signature>
</original_document>
```

(2) 封装用 xml 签名。签名必须是被签名元素的父元素。此种类型具有以下格式:

```
<Signature>
  <reference url=ID>
  </reference>
  <object ID>
    <original_document>
    ……
    </original_document>
</object>
</Signature>
```

(3) 分离的 xml 签名。签名既不是被签名数据的父元素,也不是其子元素。签名数据作为一个单独的文件而存在。此种类型具有的格式是:

```
<Signature>
  <reference url=original_document>
  </reference>
.....
</Signature>
```

本系统采用封装的 xml 签名。<cdoc>的签名项作为<cdocs>的一个子节点<signature>连到文档中。采用封装的 xml 签名的好处是签名后的文档不影响原文档的内容结构,而且直观、易于理解。

2.4 XML 签名过程

整个签名过程包括引用生成和签名生成。引用生成定义了如何计算每个<Reference>元素的摘要值,签名生成定义了如何计算实际的<SignatureValue>元素,签名生成可能包括很多引用。引用生成的最终目标是产生具有所需特征(如变换、属性和摘要值)的实际元素。同样,签名生成的最终目标是产生实际的签名值,构建全部<Signature>父元素块和所有完整的属性和子元素。签名后 Signature 节点作为<cdoc>的一个子节点连在电子病历 XML 文档中,以下是对上面关于"出院记录"的<cdoc>签名后的签名项节点内容:

```
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
    <SignedInfo>
      <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"></CanonicalizationMethod>
      <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"></SignatureMethod>
      <Reference URI="">
        <Transforms>
          <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"></Transform>
          <Transform Algorithm="http://www.w3.org/TR/1999/REC-xpath-19991116"><XPath>count(.|/descendant::extref/descendant-or-self::node())&gt;count(/descendant::extref/descendant-or-self::node())</XPath></Transform>
        </Transforms>
        <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"></DigestMethod>
        <DigestValue>y7lDUeY9tJbXf8gHO0f1+ZK0mrk=</DigestValue>
      </Reference>
    </SignedInfo>
    <SignatureValue>
                                AMVQzVQaB/vJHTgP1FY/
IcLHI5sWJijwWgbjqfRbcNJSkiC
9RY5JUmODECtUYzDWqFHgToVs
          epni2NlaLcwfUw==
    </SignatureValue>
</Signature>
```

以上签名项节点树的构架和各节点名字的含义在 XML 签名标准(RFC 3275)里都有说明,在此不赘述,只是将解释其中上面一些节点的意义。

(1) 引用结点 Reference 的 URI 属性值为空。当它的值为空时,引用结点引用的内容就是签名时签名元素(signature)所在的文档(document)。

(2) 变换过程中第一个转换节点就是用于封装的签名的语句是 <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signa-

ture" > </Transform >。这句语句的作用是将 Signature 元素从引用文档内容中剔除,因为 Signature 元素并不是待签名的内容。

(3) 第二个转换节点的作用是在签名内容中去除 < extref > 节点,这是因为在我们病历系统 DTD 中 < extref > 节点是用来链接文档外部的资源,如医学影像、声音、视频等。举个例子, < extref height = " 194" ref = " body. jpg" type = " pic" width = " 229" > </extref >。这些资源位置发生变化时, < extref > 内容必然指向另一个位置也跟着改变,但是病历内容并没发生变化。

(4) XPATH 语句的内容是: count(. |/descendant: : extref/descendant – or – self: : node( )) > count(/descendant: : extref/descendant – or – self: : node( )),请参考 XPATH 语法规则。这句 XPATH 语句的意思即是将结点名为"extref"也就是插入的图像结点从签名文档中去掉。

(5) 在这只是对病历文档的文本内容进行签名,而对链接的外部资源并没做考虑。

(6) < Signature > 中可以包含一个 < keyInfo > 项,其中记录了签名的密钥信息,供验证时获取。我们没把它放进来,验证时公钥的获取直接从证书文件中得到。

2.5 XML 签名程序

程序即是按照上面的签名过程用 java 实现,其中载入了 IBM 东京研究所开发的 XML Security Suite 关于 XML 签名的包。 (程序代码略)

3 XML 验证

3.1 验证过程

验证过程也分为引用验证和签名验证两个步骤。

引用验证首先对 < SignedInfo > 元素进行规范化。对每个要验证的 < Reference >,都要实施下述步骤:

(1) 通过对每一个 < Reference > 元素的 URI 属性引用解析来获取要待计算摘要的数据流。如果不存在任何 URI 属性,那么应用就应该知道数据源的位置。最后要待计算摘要的数据是可选的层叠变换的结果。

(2) 对第 1 个步骤中得到的数据流计算摘要,此过程使用 < DigestMethod > 元素中指定的杂凑函数来处理当前的 < Reference > 元素。

(3) 将第二步骤中计算出来的摘要值和当前正处理的 < Reference > 元素中的 < DigestValue > 元素的内容作比较,如果这些值不匹配,则引用验证失败。

签名验证的步骤:

① 从元素或具体应用中的密钥源中得到验证密钥。

② 使用的 < SignatureMethod > 规范化形式确定正使用的签名算法,并对规范化形式的 < SignedInfo > 元素计算签名值。把签名值和 < SignatureValue > 元素内的值作比较。如果这些值不匹配,则签名验证失败。

3.2 验证程序

验证程序用 java 实现,同样载入了 IBM 东京研究所开发的 XML Security Suite 包。对签名后电子病历的验证只需把解析病历 XML 文档,定位到需要验证的 < cdoc > 元素,将这个元素作为需要验证的元素通过下面的函数进行验证即可获得结果。对于签名后的文档,任何节点内容的变化都会导致验证无法通过。但根据所使用的规范化算法,节点内的空格不影响验证结果,而节点外的空格影响验证结果。(程序代码略)

4 总结

电子病历的安全在电子病历系统中占据着重要的位置。XML 签名为保证 XML 电子病历文档的完整性、不可否认性提供了一个合适的方法,而且保留了 XML 病历文档的结构化。当然病历的安全不光是这两个方面,而且病历的不可否认不仅仅是对病历编辑(修改)的操作者身份的不可否认,同时应包括操作时间的不可否认,这要通过时间戳来解决。无论怎样,这些技术的应用都要满足病历结构化的要求。

参考文献

1 Blake Dournaee 著, XML 安全基础[M], 清华大学出版社,2003。

2 Scott Oaks 著, Java 安全[M], 中国电力出版社, 2002。

3 W3C ,Donald E,Joseph M,David Solo: XML – Signature Syntax and Processing. 2002.

4 Larry Loeb :XML signatures: Behind the curtain 2001 http://www – 900. ibm. com/developerWorks/cn/xml/x – digsig/index_eng. shtml