ICS 35.040
L 80

中华人民共和国国家标准

GB/T 25061—2020
代替 GB/T 25061—2010

信息安全技术
XML 数字签名语法与处理规范

Information security technology—
XML digital signature syntax and processing specification

2020-11-19 发布　　2021-06-01 实施

国家市场监督管理总局
国家标准化管理委员会　发布

目　　次

前　言

本标准按照 GB/T 1.1—2009 给出的规则起草。

本标准代替 GB/T 25061—2010《信息安全技术　公钥基础设施 XML 数字签名语法与处理规范》，与 GB/T 25061—2010 相比，主要技术变化如下：

——增加了新的引用文件(见第 2 章)；

——在 KeyInfo 中，增加了 SM2KeyValue 类型定义，表示 SM2 椭圆曲线密码算法密钥值(见 6.5.3.3)；

——在 KeyInfo 元素中，增加了 DEREncodedKeyValue 和 KeyInfoReference 子元素，并给出模式定义(见 6.5.6 和 6.5.7)；

——增加了 xmldsig11-schema.xsd 和 xmldsig1-schema.xsd 的定义(见附录 C 中 C.2 和 C.3)；

——增加了密码杂凑算法 SM3，消息鉴别算法 HMAC-SM3，签名算法 SM2-SM3 的定义(见附录 D 中 D.3.2、D.4.3 和 D.5.3)；

——增加了 XML 规范化 1.1 算法和独占 XML 规范化 1.0 算法(见附录 D 中 D.6.3 和 D.6.4)。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别这些专利的责任。

本标准由全国信息安全标准化技术委员会(SAC/TC 260)提出并归口。

本标准起草单位：北京信安世纪科技股份有限公司、格尔软件股份有限公司、数安时代科技股份有限公司、国家密码管理局商用密码检测中心。

本标准主要起草人：汪宗斌、刘婷、郑强、张永强、吕春梅、焦靖伟、史晓峰。

本标准所代替标准的历次版本发布情况为：

——GB/T 25061—2010。

信息安全技术
XML 数字签名语法与处理规范

1 范围

本标准规定了创建和表示 XML 数字签名的处理规则、签名语法、附加的签名语法和证实方法。

本标准适用于制作和处理 XML 数字签名的应用程序、系统或服务。

2 规范性引用文件

下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件，仅注日期的版本适用于本文件。凡是不注日期的引用文件，其最新版本(包括所有的修改单)适用于本文件。

GB/T 1988 信息技术 信息交换用七位编码字符集

GB/T 13000 信息技术 通用多八位编码字符集(UCS)

GB/T 16264.8—2005 信息技术 开放系统互连 目录 第 8 部分：公钥和属性证书框架

GB/T 18793—2002 信息技术 可扩展置标语言(XML)1.0

GB/T 20518—2018 信息安全技术 公钥基础设施 数字证书格式

GB/T 35276—2017 信息安全技术 SM2 密码算法使用规范

RFC 2045 基于多用途互联网邮件扩展 第 1 部分：互联网消息体格式(Multipurpose Internet Mail Extensions(MIME) Part One: Format of Internet Message Bodies)

RFC 3279 互联网 X.509 公钥基础设施的算法和标识符 证书和证书撤销列表轮廓[Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile]

RFC 3986 统一资源标识符(URI)：通用语法[Uniform Resource Identifier (URI): Generic Syntax]

RFC 4514 轻型目录访问协议(LDAP)：甄别名的字符串表示[Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names]

RFC 5480 椭圆曲线密码主体公钥信息(Elliptic Curve Cryptography Subject Public Key Information)

3 术语、定义和缩略语

3.1 术语和定义

GB/T 18793—2002 界定的以及下列术语和定义适用于本文件。

3.1.1

分离签名 detached signature

签名于 Signature 元素以外的内容上，签名和数据对象位于不同 XML 文档中的 XML 签名文档的组织形式。

3.1.2

封内签名　enveloping signature

签名于 Signature 元素中的 Object 元素之上，以 Signature 为父元素，将原始文档包含在 Signature 中的 XML 签名文档的组织形式。

3.1.3

封皮签名　enveloped signature

签名于整个 XML 内容之上，然后将 Signature 作为子元素插入到原始文档中的 XML 签名文档组织形式。

3.1.4

签名　Signature

签名者使用私钥对待签名数据的杂凑值做密码运算得到的结果。

注：XML 签名有三种描述方式：分离签名、封内签名和封皮签名。

3.1.5

签名应用程序　signature application

实现了 Signature 元素类型的结构及其子结构的应用程序。

3.1.6

变换　transform

把一个数据从原始状态转化成导出状态的处理。

示例：XML 规范化，XPath 和 XSLT 变换等。

3.2　符号和缩略语

下列符号和缩略语适用于本文件。

?：前一符号出现 0 次或 1 次

＋：前一符号出现 1 次或多次

＊：前一符号出现 0 次、1 次或多次

CA：证书认证机构(Certificate Authority)

CRL：证书撤销列表(Certificate Revocation List)

HTTP：超文本传输协议(Hypertext Transfer Protocol)

MIME：基于多用途互联网邮件扩展(Multipurpose Internet Mail Extensions)

OID：对象标识符(Object Identifier)

PKI：公钥基础设施(Public Key Infrastructure)

URI：统一资源标识符(Universal Resource Identifier)

XML：可扩展置标语言(Extensible Markup Language)

XPath：XML 路径(XML Path)

XSL：可扩展样式表语言(Extensible Stylesheet Language)

XSLT：XSL 变换(Extensible Stylesheet Language Transformations)

4　XML 签名概述

4.1　概述

本章描述了 XML 数字签名的结构，第 5 章给出了处理规则、第 6 章签名语法和第 7 章附加的签名

语法，XML 格式要求见 GB/T 18793—2002。

XML 签名可通过间接方式作用于任意数据对象，处理的步骤是先对数据对象进行杂凑处理，处理后的结果放置在一个元素中，再对得到的元素进行杂凑处理并且通过密码学方法进行签名。XML 数字签名使用 Signature 元素来表示，其结构如下：

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

签名是通过 URI 关联数据对象的。在 XML 文档内部，签名通过 XML 片段标识符关联本地的数据对象，本地数据可包含在封内签名中，也可包含在封皮签名中。分离签名作用于外部网络资源或者作用于以兄弟元素形式出现的同一个 XML 文档的本地数据对象，因此这种签名既不是封内签名也不是封皮签名。一个 XML 文档中签名元素（以及它的 ID 和属性值和名字）可与其他元素同时存在，也可和其他元素结合在一起，命名时应注意避免违反 XML 标识的唯一性。

本标准凡涉及密码算法相关内容，按照国家有关法规实施。

签名实例参见附录 A。

4.2 定义文件用法说明

本标准附录 B 为数字签名文档类型定义，附录 C 为 XML 数字签名模式定义。

在应用本标准时，应将附录 B 和附录 C 的文件存放到应用可访问的位置，例如，假定存放在 IP 地址为 127.0.0.1 的服务器上，各个文件的路径为：http://127.0.0.1/2001/XMLSchema.dtd、http://127.0.0.1/2000/09/xmldsig-core-schema.xsd、http://127.0.0.1/2009/xmldsig11-schema.xsd、http://127.0.0.1/TR/xmldsig-core1/xmldsig1- schema.xsd，应用可根据实际情况调整存放位置，可使用附录 C 中 C.3 定义的方法来集合这些定义。

本标准提及的上述地址（“127.0.0.1”）仅是为了明确一个特定空间，实际应用中可视情况调整。

5 处理规则

5.1 生成

5.1.1 Reference 生成

对于每个要签名的数据对象，Reference 元素生成的步骤如下：

a） 根据应用程序的要求，对数据对象进行变换；

b） 计算变换后的数据对象的杂凑值；

c） 创建一个 Reference 元素，包括一个可选的数据对象的标识，可选的变换元素，密码杂凑算法和杂凑值。

5.1.2 Signature 生成

Signature 元素生成的步骤如下：

a） 以 SignatureMethod 指定的签名算法、CanonicalizationMethod 指定的规范化算法和引用生成的 Reference 为内容，创建 SignedInfo 元素。

b） 用 SingedInfo 中指定的规范化算法进行处理，并用 SignedInfo 指定的签名算法来计算 SignedInfo 的签名值。

c） 构建包括 SignedInfo、Object、KeyInfo 和 SignatureValue 的 Signature 元素。Signature 元素中各个子元素的含义和具体构造方法见第 6 章。

5.2 确认

5.2.1 概述

确认应包括：

a） 引用确认，验证 SignedInfo 中每个 Reference 包含的杂凑值；

b） 签名确认，使用密码方法对计算 SignedInfo 得到的签名进行签名确认。

5.2.2 Reference 确认

引用确认的步骤如下：

a） 根据 SignedInfo 中 CanonicalizationMethod 指定的规范化方法来处理 SignedInfo 元素；

b） 对于 SignedInfo 中的每个 Reference：

1） 获得进行杂凑处理的数据对象；

2） 使用 Reference 中指定的密码杂凑算法对结果数据对象计算出杂凑值；

3） 将上一步生成的杂凑值和 SignedInfo 中的 DigestValue 元素的值进行比较，如果有不同，那么确认失败。

注：SignedInfo 在步骤 a)进行了规范化，应用程序宜确保 CanonicalizationMethod 不会产生错误。

5.2.3 Signature 确认

Signature 确认应比较以 CanonicalizationMethod 指定的规范化方法和 SignatureMethod 指定的签名方法处理 SignedInfo 的结果是否与 SignatureValue 中的值是否匹配。

Signature 确认的步骤如下：

a） 从 KeyInfo 或者外部源获得密钥信息；

b） 使用 CanonicalizationMethod 来获得 SignatureMethod 的规范化形式，然后用得出的结果和上面得到的密钥信息对 SignedInfo 元素进行签名值验证。

6 签名语法

6.1 概述

6.1.1 模式定义

签名语法通过 XML 模式定义来定义，所有的 XML 模式定义使用下面的 XML 前导说明部分、文

件类型声明和内部实体。

模式定义：

```
<? xml version="1.0" encoding="UTF-8"? >
```

注：上一行为 XML 声明，该行中的<? xml 是一个整体，表示是 XML 文件的开始，而本标准 3.2 中定义的“?”则表示元素的个数，请注意区分。

```
<! DOCTYPE schema
PUBLIC "-//W3C//DTD XMLSchema 200102//EN" "http://127.0.0.1/2001/XMLSchema.dtd"
[
<! ATTLIST schema
xmlns:ds CDATA #FIXED "http://127.0.0.1/2000/09/xmldsig#">
<! ENTITY dsig 'http://127.0.0.1/2000/09/xmldsig#'>
<! ENTITY % p ''>
<! ENTITY % s ''>
]>

<schema          xmlns="http://127.0.0.1/2001/XMLSchema"
xmlns:ds="http://127.0.0.1/2000/09/xmldsig#"
targetNamespace="http://127.0.0.1/2000/09/xmldsig#"
version="0.1" elementFormDefault="qualified">
```

文档类型定义：

```
<! ENTITY % Object.ANY ''>
<! ENTITY % Method.ANY ''>
<! ENTITY % <Transform>.ANY ''>
<! ENTITY % SignatureProperty.ANY ''>
<! ENTITY % KeyInfo.ANY ''>
<! ENTITY % KeyValue.ANY ''>
<! ENTITY % X509Data.ANY ''>
```

扩展标记使用 dsig11：名字空间。新的模式定义如下：

```
<? xml version="1.0" encoding="utf-8"? >
<schema xmlns="http://127.0.0.1/2001/XMLSchema"
        xmlns:ds="http://127.0.0.1/2000/09/xmldsig#"
        xmlns:dsig11="http://127.0.0.1/2009/xmldsig11#"
        targetNamespace="http://127.0.0.1/2009/xmldsig11#"
        version="0.1" elementFormDefault="qualified">
```

6.1.2 ds：CryptoBinary 简单类型

定义 ds：CryptoBinary 简单类型，把 XML 中任意长度的整数表示成字节字符串。具体方法是先把整数值转化成高位在前格式的位串，在位串前面补 0 使得位数是 8 的整数倍，去掉开头为零字节（连续 8 个 0 的位串），然后对这个字节串进行 base64 编码，base64 编码遵循 RFC 2045。

注：base64Binary 与 CryptoBinary 类型相同，定义一个新的类型主要是兼容不同的使用习惯。

模式定义：

```
<simpleType name="CryptoBinary">
```

```
        <restriction base="base64Binary">
        </restriction>
    </simpleType>
```

6.2 Signature 元素

Signature 元素是 XML 签名的根元素，Signature 元素的组织应遵循下面说明的模式。

模式定义：

```
<element name="Signature" type="ds:SignatureType"/>
<complexType name="SignatureType">
  <sequence>
    <element ref="ds:SignedInfo"/>
    <element ref="ds:SignatureValue"/>
    <element ref="ds:KeyInfo" minOccurs="0"/>
    <element ref="ds:Object" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
  <attribute name="Id" type="ID" use="optional"/>
</complexType>
```

文档类型定义：

```
<! ELEMENT Signature (SignedInfo, SignatureValue, KeyInfo?, Object * ) >
<! ATTLIST Signature
    xmlns CDATA #FIXED 'http://127.0.0.1/2000/09/xmldsig#'
    Id    ID            #IMPLIED >
```

6.3 SignatureValue 元素

SignatureValue 元素包含了数字签名的具体值，通常使用 base64 进行编码。当给出两个 SignatureMethod 算法时，一个是应实现的，另一个是可选实现的，用户可使用自己定义的算法。

注：封皮签名在计算 SignatureValue 时不包含其自身。

模式定义：

```
<element name="SignatureValue" type="ds:SignatureValueType"/>
<complexType name="SignatureValueType">
  <simpleContent>
    <extension base="base64Binary">
      <attribute name="Id" type="ID" use="optional"/>
    </extension>
</simpleContent>
</complexType>
```

文档类型定义：

```
<! ELEMENT SignatureValue (#PCDATA) >
<! ATTLIST SignatureValue
          Id   ID        #IMPLIED>
```

6.4 SignedInfo 元素

6.4.1 概述

SignedInfo 的结构包括规范化算法、签名算法和一个或者多个引用。SignedInfo 元素可包含一个可选的 ID 属性，供其他签名或者对象引用。

SignedInfo 不包括显式的签名或杂凑属性(例如处理时间、加密设备序列号等)，如果应用程序需要将属性与签名或杂凑相关联，则可在 Object 元素内的 SignatureProperties 元素中包含此类信息。

模式定义：

```
<element name="SignedInfo" type="ds:SignedInfoType"/>
<complexType name="SignedInfoType">
  <sequence>
    <element ref="ds:CanonicalizationMethod"/>
    <element ref="ds:SignatureMethod"/>
    <element ref="ds:Reference" maxOccurs="unbounded"/>
  </sequence>
  <attribute name="Id" type="ID" use="optional"/>
</complexType>
```

文档类型定义：

```
<! ELEMENT SignedInfo (CanonicalizationMethod,
SignatureMethod, Reference+) >
<! ATTLIST SignedInfo
  Id   ID        #IMPLIED>
```

6.4.2 CanonicalizationMethod 元素

CanonicalizationMethod 是 SignedInfo 元素中用于指定规范化算法的必要元素，指明签名处理之前进行规范化处理的算法，CanonicalizationMethod 元素使用算法标识符和实现需求中给出的算法。应用实现应支持必要的规范化算法。

可选用需要的规范化算法，若不明确指定，缺省的规范化算法是 Canonical XML。

对 SignedInfo 元素的呈现与规范化算法本身有关。下面的步骤适用于处理 XML 节点的算法：

基于 XML 的规范化实现，实现带有一个 XPath 节点集合，节点集合源于包含 SignedInfo 元素的文档，并指明当前的 SignedInfo，后代、属性、SignedInfo 名字空间节点和它的后代元素。

模式定义：

```
<element name="CanonicalizationMethod" type="ds:CanonicalizationMethodType"/>
<complexType name="CanonicalizationMethodType" mixed="true">
    <sequence>
        <any namespace="##any" minOccurs="0" maxOccurs="unbounded"/>
        <! -- (0,unbounded) elements from (1,1) namespace -->
    </sequence>
    <attribute name="Algorithm" type="anyURI" use="required"/>
</complexType>
```

文档类型定义：

```
<! ELEMENT CanonicalizationMethod (#PCDATA %Method.ANY;)* >
<! ATTLIST CanonicalizationMethod
    Algorithm CDATA #REQUIRED>
```

6.4.3 SignatureMethod 元素

SignatureMethod 是用来指定进行签名生成和验证算法的必要元素，表明签名操作中用到的密码函数(如密码杂凑算法，公钥算法，MAC，填充方式等)。这个元素使用算法标识符(算法标识符实例可参考附录 D)。

模式定义：

```
<element name="SignatureMethod" type="ds:SignatureMethodType"/>
<complexType name="SignatureMethodType" mixed="true">
  <sequence>
    < element  name = " HMACOutputLength "  type = " ds: HMACOutputLengthType "
minOccurs="0" />
    <any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
    <! -- (0,unbounded) elements from (1,1) external namespace -->
  </sequence>
  <attribute name="Algorithm" type="anyURI" use="required"/>
</complexType>
```

文档类型定义：

```
<! ELEMENT SignatureMethod (#PCDATA|HMACOutputLength % Method.ANY;)* >
<! ATTLIST SignatureMethod
    Algorithm CDATA #REQUIRED >
```

6.4.4 Reference 元素

6.4.4.1 概述

Reference 元素可出现一次或者多次，用来指明密码杂凑算法、杂凑值、签名对象的标识符、签名对象的类型和进行杂凑处理前的一个变换列表。标识(URI)和变换描述了如何对内容进行杂凑处理。Type 属性指明如何处理引用的数据。可选的 ID 属性允许 Reference 引用其他的内容。

模式定义：

```
<element name="Reference" type="ds:ReferenceType"/>
<complexType name="ReferenceType">
  <sequence>
    <element ref="ds:Transforms" minOccurs="0"/>
    <element ref="ds:DigestMethod"/>
    <element ref="ds:DigestValue"/>
  </sequence>
  <attribute name="Id" type="ID" use="optional"/>
  <attribute name="URI" type="anyURI" use="optional"/>
  <attribute name="Type" type="anyURI" use="optional"/>
</complexType>
```

文档类型定义：

```
<! ELEMENT Reference (Transforms?, DigestMethod, DigestValue) >
<! ATTLIST Reference
  Id        ID        #IMPLIED
  URI       CDATA     #IMPLIED
  Type      CDATA     #IMPLIED>
```

6.4.4.2 URI 属性

URI 属性使用 URI 引用来标识一个数据对象，URI 和 XML 均使用 GB/T 13000 定义的字符集。但 URI 引用中禁止使用除＃，%，[，]外，RFC 3986 中列出的所有非 GB/T 1988 字符和保留字等特定字符。禁止使用的字符应通过下面的方法进行转义：

a） 每个禁止字符按照一个或多个字节转化成 GB/T 13000 编码；

b） 任何与一个禁止字符相应的字节序列使用 URI 的转义方法进行转义；

c） 用生成的字符序列代替原始的字符。

XML 签名的应用程序应能解析 URI 语法，并能够依据 HTTP 标准来解析 URI 引用、理解协议的参数和状态信息，处理 HTTP 的状态码。

一个资源有多个 URI 标识时，应使用最具体的 URI 标识。

应用程序预先知道对象的标识时，可不提供 URI 属性。

URI 的 Type 属性包含被签名对象的类型信息，表示成 URI，Type 属性是可选的。

例如：

Type="http://127.0.0.1/2000/09/xmldsig＃Object"，或

Type="http://127.0.0.1/2000/09/xmldsig＃Manifest"

Type 属性应指向具体的对象，而不是其内容，Type 属性是辅助性的，不要求验证 Type 的有效性。

6.4.4.3 Reference 处理模型

签名应用程序不必为了符合本标准而与 XPath 规范一致，但对于那些希望充分利用 XML 特性，将 XML 签名生成作为应用程序的一部分来处理的应用程序则应使用 XPath 数据模型、定义和语法。采用 XPath 的目的是为那些希望使用这些特征，而又符合 XPath 规范的应用提供一种可选途径。应用可对一个节点集合进行充分的功能替换，并且仅实现本标准需要的那些 Xpath 表达式行为，为了简单起见，本标准通常使用 Xpath 术语而不在每个地方都注明。对于“XPath 节点集合”需求可实现一个包含节点集合功能相同的应用程序。应用程序应对 XML 文档采用与 Xpath 处理等效的方式处理文档。

解析 URI 或者一系列 Transform 的变换结果的数据类型是一个八位位组流或者是一个 Xpath 的节点集合。

本标准中所涉及的变换是根据输入定义的。签名应用程序的正常行为应为：

a） 如果数据对象是八位位组流且下一个变换要求一个节点集合，签名应用程序应对字节流进行分析，通过 XML 标准处理过程来得出必需的节点集合；

b） 如果数据对象是一个节点集合而下一个变换需要八位位组，签名应用程序必须使用规范的 XML 来把节点集合转化成八位位组流。

在需要不同输入的变换中进行变换时，用户可指定替代的变换来覆盖这些缺省的变换，最终的八位位组流包含了受保护的数据，用 DigestMethod 指定的密码杂凑算法对这些数据对象进行处理，得出的结果放在 DigestValue 中。

若 URI 引用为非同文档引用，解析 URI 引用的结果应为一个八位位组流。URI 标识的 XML 文档指向同文档引用或应用不要求变换时，签名应用程序不必解析它。

若片段出现在一个绝对或者相对 URI 的前面，那么片段的含义由资源的 MIME 类型定义。对于 XML 文档，也可通过一个代理来完成签名程序对 URI 的解析(包括对片段的处理)。如果片段处理不是标准化处理的引用确认可能会失败。本标准建议 URI 属性不包括片段标识符，而将这种处理过程当作附加的 XPath 变换来进行说明。

当片段没有出现在 URI 前面时，XML 签名程序应支持空 URI 和无名 Xpointer；若应用程序还要支持任何保存注释的规范化操作，那么建议对同文档的 Xpointer 提供支持。由于应用程序可能无法控制片段的生成，因此所有其他对 Xpointer 的支持都是可选的，所有对无名和外部引用的 Xpointer 的支持也是可选的。

6.4.4.4 同文档 URI 引用

解析同文档引用应产生一个适合规范化 XML 使用的 Xpath 节点集合。特别是，解析一个空 URI 应产生一个 Xpath 节点集合，该集合包含拥有 URI 属性的 XML 文档的每个非注释节点。片段 URI 中＃号后面的字符应符合 Xpointer 语法，处理 Xpointer 的时候，应用程序应使用包含 URI 属性的 XML 文档的根节点来初始化 Xpointer 处理文档。若 Xpointer 处理后的结果是一个节点集合时，应用程序应通过下面的方法获得：

a) 丢弃点节点；

b) 名字空间子资源中包括完整或者部分内容的 XPath 节点；

c) 用子节点替换根节点(假设它在节点集合里面)；

d) 把所有元素节点 E 替换为 E 和 E 的后代节点(文本、注释、PI 或者元素)以及所有的 E 和它的后代元素的名字空间和属性节点；

e) 如果 URI 不是一个完整的 XPointer，那么删除所有的注释节点。

解析时应执行步骤 d)的替换。XPointer 是使用子树的根节点元素来指明一个 XML 文档的语法分析树的子树，而规范化的 XML 把一个节点集合当作节点的集合，在这种情况下，缺少后代节点就会导致在规范形式的内容的不足。

步骤 e)用来处理空 URI、裸名指针和子序列 XPointers。当传递一个节点集合时，需要按照有注释和没注释对节点集合进行处理，处理成字节流时会调用 Xpath 表达式(缺省的或者没有注释的)，因此，为了在传递节点集合的时候保留脱模注释的缺省的行为，应去除非完整的 XPointer 的 URI。若要在通过标识符 ID 选择元素的时候保留注释，应使用这样的全 XPointer：URI＝'＃xpointer(id('ID'))'；若要在选择整个文档的时候保留注释，应使用这样的全 XPointer：URI＝'＃xpointer(/)'，XPointer 包含了一个含有根节点的简单的 Xpath 表达式，步骤 d)会将该根节点替换为语法分析树的所有节点(所有的后代、所有的属性和所有节点的名字空间)。

6.4.4.5 Transforms 元素

可选的 Transforms 元素包括一个有序的 Transform 元素列表，这些元素描述签名者如何获得将要进行杂凑处理的数据对象，每个 Transform 的输出都要作为下一个 Transform 的输入，第一个 Transform 的输入是解析 Reference 元素的 URI 属性所得到的结果，最后一个 Transform 的结果是 DigestMethod 算法的输入。使用 Transform 后，签名者处理的已经不是原始的文档，而是 Transform 处理后的文档。

每个 Transform 元素由一个算法属性和与这个算法配套的内容参数构成(如果有参数的话)，算法

属性值指定要进行处理的算法的名字，Transform 内容提供附加的数据来控制算法处理 Transform 输入的过程。

在 Reference 处理模型中曾经提到，一些 Transform 使用 XPath 节点集合为输入，而其他一些需要一个字节流。若实际的输入符号 Transform 的需求，则 Transform 在操作的时候不会更改输入。若 Transform 的输入要求和实际输入的格式不同，则实际的输入需要进行一些变换。Transform 可能需要显式的 MIME 类型、字符集或者是它们从前面的 Transform 或源数据收到的数据的相关信息，应以 Transform 算法参数的形式提供这种数据的特征，通过参数的形式把这些数据特征提供给 Transform 算法，且应在算法的规范中描述出来。

Transform 例子包括但不仅限于 base64 解码，XML 规范化，XPath 过滤和 XSLT。

模式定义：

```
<element name="Transforms" type="ds:TransformsType"/>
<complexType name="TransformsType">
  <sequence>
    <element ref="ds:Transform" maxOccurs="unbounded"/>
  </sequence>
</complexType>

<element name="Transform" type="ds:TransformType"/>
<complexType name="TransformType" mixed="true">
  <choice minOccurs="0" maxOccurs="unbounded">
    <any namespace="##other" processContents="lax"/>
    <!-- (1,1) elements from (0,unbounded) namespaces -->
    <element name="XPath" type="string"/>
  </choice>
  <attribute name="Algorithm" type="anyURI" use="required"/>
</complexType>
```

文档类型定义：

```
<! ELEMENT Transforms (<Transform>+)>
<! ELEMENT <Transform> (#PCDATA|XPath %<Transform>.ANY;)* >
<! ATTLIST <Transform>
  Algorithm     CDATA     #REQUIRED >
<! ELEMENT XPath (#PCDATA) >
```

6.4.4.6 DigestMethod 元素

DigestMethod 是指定签名对象的密码杂凑算法的必要元素，使用通用结构来描述算法标识符和实现算法(可参考附录 D)。

如果 URI 解析的结果和变换处理的结果是一个 XPath 节点集合，则结果应按 Reference 处理模型中规定的方法进行变换；如果 URI 解析和变换处理结果是字节流，则不需要进行变换，直接对得出的字节流数据进行密码杂凑算法处理。

模式定义：

```
<element name="DigestMethod" type="ds:DigestMethodType"/>
```

```
<complexType name="DigestMethodType" mixed="true">
  <sequence>
  <any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
  <attribute name="Algorithm" type="anyURI" use="required"/>
</complexType>
```

文档类型定义：

```
<! ELEMENT DigestMethod (#PCDATA %Method.ANY;)* >
<! ATTLIST DigestMethod
  Algorithm      CDATA   #REQUIRED >
```

6.4.4.7 DigestValue 元素

DigestValue 元素包含了杂凑值编码后的结果，杂凑值应使用 base64 进行编码。

模式定义：

```
<element name="DigestValue" type="ds:DigestValueType"/>
<simpleType name="DigestValueType">
    <restriction base="base64Binary"/>
</simpleType>
```

文档类型定义：

```
<! ELEMENT DigestValue (#PCDATA) >
<! -- base64 encoded digest value -->
```

6.5 KeyInfo 元素

6.5.1 概述

KeyInfo 是接收者获取确认签名所需密钥材料的可选元素。KeyInfo 可包括的信息有：密钥、名字、证书和其他公开密钥管理信息。本标准定义了一些简单类型，应用程序可扩展这些类型，也可用 XML 名字空间定义自己的密钥标识和交换语义来代替它们。但密钥信息的可信问题(例如，它的真实性或强度)不在本标准讨论范围之内，而应由应用程序来处理。

接收者能从应用程序的上下文中获取验证签名的密钥时可忽略 KeyInfo 元素，KeyInfo 中的多个声明可指向同一个密钥，应用程序应实现 KeyValue，但不一定实现 RetrievalMethod。应用也可通过引入不同名字空间的元素，来定义和使用自己选择的任意机制。

KeyInfo 的子元素(例如 X509Data)的模式/文档类型定义规范允许其内容被其他命名空间的元素扩展，但仅在忽略扩展元素时仍然安全才有效，否则扩展元素(包括元素的替代结构)应为 KeyInfo 的子元素。

下面列出了 disg：名字空间中已分配标识符的 KeyInfo 类型，可在 RetrievalMethod 元素的 Type 属性中使用它们来描述一个远程的 KeyInfo 结构。

——http://127.0.0.1/2000/09/xmldsig#RSAKeyValue

——http://127.0.0.1/2000/09/xmldsig#X509Data

下面列出了在 dsig11：名字空间中分配标识符的其他 KeyInfo 类型。

http://127.0.0.1/2009/xmldsig11#SM2KeyValue

http://127.0.0.1/2009/xmldsig11#DEREncodedKeyValue

除了上面用 XML 结构定义的这些类型，本标准定义了一个类型来表示二进制的数字证书（又称 X509 证书），具体定义见 GB/T 16264.8—2005 的第 7 章。

http://127.0.0.1/2000/09/xmldsig#rawX509Certificate

模式定义：

```
<element name="KeyInfo" type="ds:KeyInfoType"/>
<complexType name="KeyInfoType" mixed="true">
  <choice maxOccurs="unbounded">
    <element ref="ds:KeyName"/>
    <element ref="ds:KeyValue"/>
    <element ref="ds:RetrievalMethod"/>
    <element ref="ds:X509Data"/>
    <! -- <element ref="dsig11:DEREncodedKeyValue"/> -->
    <! -- DEREncodedKeyValue (XMLDsig 1.1) will use the any element -->
    <! -- <element ref="dsig11:KeyInfoReference"/> -->
    <! -- KeyInfoReference (XMLDsig 1.1) will use the any element -->
    <any processContents="lax" namespace="##other"/>
    <! -- (1,1) elements from (0,unbounded) namespaces -->
  </choice>
    <attribute name="Id" type="ID" use="optional"/>
</complexType>
```

文档类型定义：

```
<! ELEMENT KeyInfo (#PCDATA|KeyName|KeyValue|RetrievalMethod|X509Data %KeyInfo.ANY;)* >
<! ATTLIST KeyInfo
    Id  ID  #IMPLIED >
```

6.5.2 KeyName 元素

KeyName 元素应包含一个字符串（其中的空格是不能忽略的），签名者可用它与接收者传递密钥标识符。KeyName 应包含签名的密钥对的相关标识符，也可包括对其他与协议相关的间接标识密钥对的信息。

模式定义：

```
<element name="KeyName" type="string"/>
```

文档类型定义：

```
<! ELEMENT KeyName (#PCDATA) >
```

6.5.3 KeyValue 元素

6.5.3.1 概述

KeyValue 应包含唯一有效确认签名的公钥。KeyValue 元素可包括在外部定义的公钥值，它们以 PCDATA 或外部名字空间的元素类型来表示。在附录 A 中给出了 SM2 公开密钥的结构化格式的实例。

模式定义：

```
<element name="KeyValue" type="ds:KeyValueType"/>
<complexType name="KeyValueType" mixed="true">
    <choice>
        <element ref="dsig11:SM2KeyValue"/>
        <element ref="ds:RSAKeyValue"/>
        <any namespace="##other" processContents="lax"/>
    </choice>
</complexType>
```

文档类型定义：

```
<! ELEMENT KeyValue (#PCDATA| RSAKeyValue %KeyValue.ANY;)* >
```

6.5.3.2 RSAKeyValue 元素

标识符：

Type="http://127.0.0.1/2000/09/xmldsig#RSAKeyValue"

（可用在 RetrievalMethod 或者 Reference 元素中用来表示类型）

RSA 密钥值有两个域：模和指数

实例：

```
<RSAKeyValue>
    <Modulus>xA7SEU+e0yQH5rm9kbCDN9o3aPIo7HbP7tX6WOoc
    LZAtNfyxSZDU16ksL6WjubafOqNEpcwR3RdFsT7bCqnXPBe5E
    Lh5u4VEy19MzxkXRgrMvavzyBpVRgBUwUlV5foK5hhmbktQhy
    Ndy/6LpQRhDUDsTvK+g9Ucj47es9AQJ3U=
    </Modulus>
    <Exponent>AQAB</Exponent>
</RSAKeyValue>
```

任意长度的整数应遵循 XML 中 ds:CryptoBinary 类型的定义，用 XML 表示为字节串。

模式定义：

```
<element name="RSAKeyValue" type="ds:RSAKeyValueType"/>
<complexType name="RSAKeyValueType">
  <sequence>
    <element name="Modulus" type="ds:CryptoBinary"/>
    <element name="Exponent" type="ds:CryptoBinary"/>
  </sequence>
</complexType>
```

文档类型定义：

```
<! ELEMENT RSAKeyValue (Modulus, Exponent) >
<! ELEMENT Modulus (#PCDATA) >
<! ELEMENT Exponent (#PCDATA) >
```

6.5.3.3 SM2KeyValue 元素

标识符：

Type="http://127.0.0.1/2009/xmldsig11#SM2KeyValue"

(可用在 RetrievalMethod 或者 Reference 元素中用来表示类型)

SM2KeyValue 元素定义在 http://127.0.0.1/2009/xmldsig11#名字空间中。

SM2 公钥由域参数和 PublicKey 两部分组成。

实例:

```
< SM2KeyValue xmlns="http://127.0.0.1/2009/xmldsig11#">
    <NamedCurve URI="urn:oid:1.2.156.10197.1.301" />
    <PublicKey> BDanuWjsEP0Ki2ursklRv+pBeXNnGFkrS2Veu/FrudFKFTiscMJ1TCa0K23Lk0
+L7HIGz+SWFZZjDAnHPD7n+YQ </PublicKey>
</SM2KeyValue>
```

域参数可通过引用使用 dsig11:NamedCurve 元素编码。命名曲线由 URI 属性指定,通过 OID 定义。程序应支持 dsig11:NamedCurve 元素和 OID 为 1.2.156.10197.1.301 的 256 位素域椭圆曲线。

PublicKey 元素包含该点 x 和 y 坐标二进制的 Base64 编码。其值计算方法如下:

a) 首先将字段元素 x 和 y 转换为八位字符串,然后将椭圆曲线点(x,y)转换为八位字节字符串,再将转换结果前面加上 0x04,OID 为 1.2.156.10197.1.301 时公钥的 x 分量和 y 分量的长度为 256 比特;

b) Base64 对步骤 a)中的转换所产生的八位组串进行编码。

模式定义:

```
<! -- targetNamespace="http://127.0.0.1/2009/xmldsig11#" -->
<element name="SM2KeyValue" type="dsig11: SM2KeyValueType" />
<complexType name=" SM2KeyValueType">
    <sequence>
        <element name="NamedCurve" type="dsig11:NamedCurveType" />
        <element name="PublicKey" type="dsig11:ECPointType" />
    </sequence>
    <attribute name="Id" type="ID" use="optional" />
</complexType>

<complexType name="NamedCurveType">
            <attribute name="URI" type="anyURI" use="required" />
</complexType>

<simpleType name="ECPointType">
            <restriction base="ds:CryptoBinary" />
</simpleType>
```

文档类型定义:

```
<! ELEMENT SM2KeyValue (NamedCurve, PublicKey) >
<! ELEMENT NamedCurve (#PCDATA) >
<! ELEMENT PublicKey (#PCDATA) >
```

6.5.4 RetrievalMethod 元素

KeyInfo 中的 RetrievalMethod 元素用于传递存储在另一个位置的 KeyInfo 信息的引用。例如，一个文档内的几个签名使用一个内部或者外部的 X509 证书链来验证签名，每个签名的 KeyInfo 都可用一个 RetrievalMethod 元素来引用证书链，而无须每次都使用 X509Certificate 元素的完整证书链。

除了没有 DigestMethod 或 DigestValue 子元素，RetrievalMethod 使用与 Reference 的 URI 属性和 Reference 处理模型同样的语法和解析引用的行为，且应包含 URI。

Type 是表示要检索的数据类型的可选标识符。本标准定义了具有相应 XML 结构的 KeyInfo 类型，解析一个 RetrievalMethod 的 Reference 的结果是一个 XML 元素或者是以此元素为根的文档。KeyInfo 的 rawX509Certificate 类型(不包含 XML 结构)返回一个二进制 X509 证书。

模式定义：

```
<element name="RetrievalMethod" type="ds:RetrievalMethodType"/>
<complexType name="RetrievalMethodType">
    <sequence>
        <element ref="ds:Transforms" minOccurs="0"/>
    </sequence>
    <attribute name="URI" type="anyURI"/>
    <attribute name="Type" type="anyURI" use="optional"/>
</complexType>
```

文档类型定义：

```
<! ELEMENT RetrievalMethod (Transforms?) >
<! ATTLIST RetrievalMethod
    URI     CDATA           #REQUIRED
    Type    CDATA           #IMPLIED >
```

6.5.5 X509Data 元素

标识符：

Type="http://127.0.0.1/2000/09/xmldsig#X509Data"

(在 RetrievalMethod 或者 Reference 元素里用来表示类型)

KeyInfo 中的 X509Data 元素包括一个或多个密钥标识符，或 X509 证书(证书标识符或废止列表)。X509Data 的内容有：

a) 至少应使用一个下列元素类型中的元素，当描述或者关联同一个证书时，可一同使用或多次使用下列元素；

b) 具体的内容如下：

- X509IssuerSerial 元素，应包含一个符合 RFC 4514 的 X509issuer 的甄别名/序列号对，甄别名的生成应当符合甄别名编码规则。
- X509SubjectName 元素，应包含一个 X509 证书主体名字，应遵循 RFC 4514。
- X509SKI 元素，应包含 X509 证书主体密钥标识符扩展的 base64 简单编码。
- X509Certificate 元素，应包含一个 base64 编码的 X509 证书。
- X509CRL 元素，应包含一个 base64 编码的证书撤销列表(CRL)。
- dsig11：X509Digest 元素，应包含一个 base64 编码的证书杂凑值。该元素应包含

Algorithm 属性标识的密码杂凑算法的 URI，杂凑的输入应为证书的原始八位字节串。

- 伴随或补充上述元素的外部名字空间的元素。

X509IssuerSerial，X509SKI，X509SubjectName 和 dsig11：X509Digest 元素应指向证书或者包含确认密钥的证书。所有指向一个特定的证书的元素应放在 X509Data 元素中，而且引用应指向这个 X509Data 元素。

同一个密钥而和不同证书关联的 X509IssuerSerial，X509SKI，X509SubjectName 和 dsig11：X509Digest 元素应分组到一个 KeyInfo 元素中，但是可出现在多个 X509Data 元素中。

出现在 X509Data 元素中的证书应能关联到确认密钥，可通过包含确认密钥，或作为证书链的一部分包含这个确认密钥，但不要求证书链有序。

下面是具体的例子：

```
<KeyInfo>
  <X509Data> <! -- two pointers to certificate-A -->
      <X509IssuerSerial>
          <X509IssuerName>CN=Zongbin WANG, OU=R&D, O=INFOSEC,
              L=Haidian District, ST=Beijing, C=CN</X509IssuerName>
          <X509SerialNumber>12345678</X509SerialNumber>
      </X509IssuerSerial>
      <X509SKI>31d97bd7</X509SKI>
</X509Data>
<X509Data> <! -- single pointer to certificate-B -->
      <X509SubjectName>Subject of Certificate B</X509SubjectName>
</X509Data>
<X509Data> <! -- certificate chain -->
      <! --Signer cert, issuer CN=arbolCA,OU=R&D,O=INFOSEC,C=CN, serial 4-->
      <X509Certificate>MIICXTCCA..</X509Certificate>
      <! -- Intermediate cert subject CN=arbolCA,OU=R&D,O=INFOSEC,C=CN
          issuer CN=tootiseCA,OU=R&D,O=INFOSEC,C=CN -->
      <X509Certificate>MIICPzCCA...</X509Certificate>
      <! -- Root cert subject CN=tootiseCA,OU=R&D,O=INFOSEC,C=CN -->
      <X509Certificate>MIICSTCCA...</X509Certificate>
  </X509Data>
</KeyInfo>
```

注：对于 PKCS# 7 编码的证书链或者 CRL 没有直接的规定。在一个<X509Data>元素中可出现一组证书和 CRL，而且在一个 KeyInfo 元素中可出现多个 X509Data 元素。每当一个<X509Data>元素中出现多个证书时，其中一个证书包含验证签名的公钥。

DN 中的字符串(<X509IssuerSerial>、<X509SubjectName>或<KeyName>)应按照下述方法编码：

- 把字符串当作连续的 GB/T 13000 字符串；
- 特殊字符应加上“\”来转义，特殊字符包括字符串的最前面“#”或字符中的“,”“+”“"”“\”“<”“>”或者“;”；
- 转义所有的 GB/T 1988 控制字符(GB/T 13000 范围内的\x00～\x1f)，即在它们的 GB/T 13000两位 16 进制数之前加上“\”字符；

- 转义所有的后续的空格，用“\20”代替“\ ”。

因 XML 文档逻辑上包含字符，而不是字节，故应根据产生该 XML 文档的物理表示所使用的字符编码方法来对结果 GB/T 13000 字符串进行编码。

引入 dsig11：X509Digest 元素后，可弃用 X509IssuerSerial 元素。

6.5.6 DEREncodedKeyValue 元素

标识符：

Type="http://127.0.0.1/2009/xmldsig11#DEREncodedKeyValue"

（可在 RetrievalMethod 或 Reference 元素中使用，来识别指示对象的类型）

X.509 证书的主体公钥信息字段中公钥算法和值，编码的要求见 GB/T 20518—2018 中 5.2.3.7，编码后再进行 base64 简单编码。

对于本标准中支持的密钥类型，下面的标准文件标识了密钥/算法类型的主体公钥信息格式和相关 OID 值：

——SM2：见 GB/T 35276—2017 中 7.1；

——RSA：见 RFC 3279；

——EC：见 RFC 5480。

自定义扩展密钥类型可见下面的方式。

模式定义：

```
<! -- targetNamespace="http://127.0.0.1/2009/xmldsig11#" -->
<element name="DEREncodedKeyValue" type="dsig11:DEREncodedKeyValueType" />
  <complexType name="DEREncodedKeyValueType">
    <simpleContent>
      <extension base="base64Binary">
        <attribute name="Id" type="ID" use="optional"/>
      </extension>
    </simpleContent>
  </complexType>
```

6.5.7 KeyInfoReference 元素

KeyInfo 中的 KeyInfoReference 元素用于传递 KeyInfo 元素的位置引用。例如，文档中的多个签名可能使用同一个证书链验证的密钥。每个签名的 KeyInfo 可使用一个 KeyInfoReference 元素来引用这个证书链，而不是证书链包含多个 X509Certificate 元素的序列。

KeyInfoReference 的使用与 Reference 的 URI 属性和 Reference 处理模型相同的语法，但没有子元素且要求使用 URI 属性。

KeyInfoReference 取得的结果应为一个 KeyInfo 元素，或者一个以 KeyInfo 为根元素的 XML 文档。

模式定义：

```
<! -- targetNamespace="http://127.0.0.1/2009/xmldsig11#" -->
<element name="KeyInfoReference" type="dsig11:KeyInfoReferenceType"/>
<complexType name="KeyInfoReferenceType">
  <attribute name="URI" type="anyURI" use="required"/>
```

```
    <attribute name="Id" type="ID" use="optional"/>
</complexType>
```

6.6 Object 元素

标识符：

Type=http://127.0.0.1/2000/09/xmldsig#Object（可用在 Reference 元素中表示类型。）

Object 是可包含任何数据的可选元素，可出现一次或多次。Object 元素可包括可选的 MIME 类型、ID 和编码属性。

Object 的编码属性可用 URI 方式标识 Object 所用编码(例如一个二进制文件)。

MimeType 可选属性是描述 Object 中数据编码的字符串值，具体的类型定义见 RFC 2045，此属性纯粹是辅助性的，本标准不要求对 MimeType 信息进行验证，应用程序应处理标准类型编码和签名确认编码的变换。例如，如果对象中包括 base64 编码的 PNG，那么 Encoding 可指定为 base64 并且 MimeType 指定为“image/png”。

SignedInfo 或 Manifest 通过 Reference 来引用 Object 的 Id，Object 元素常用于封内签名，被签名的对象将成为签名元素的一部分。对整个 Object 元素计算杂凑值，包括开始和结束标签。

模式定义：

```
<element name="Object" type="ds:ObjectType"/>
<complexType name="ObjectType" mixed="true">
    <sequence minOccurs="0" maxOccurs="unbounded">
        <any namespace="##any" processContents="lax"/>
    </sequence>
    <attribute name="Id" type="ID" use="optional"/>
    <attribute name="MimeType" type="string" use="optional"/>
    <attribute name="Encoding" type="anyURI" use="optional"/>
</complexType>
```

文档类型定义：

```
<! ELEMENT Object (#PCDATA|Signature|SignatureProperties|Manifest %Object.ANY;)
* >
<! ATTLIST Object
  Id          ID        #IMPLIED
  MimeType    CDATA     #IMPLIED
  Encoding    CDATA     #IMPLIED >
```

7 附加的签名语法

7.1 概述

下面描述 Manifest 和 SignatureProperties 元素，并描述如何处理 XML 处理指令和注释。这些元素可出现在上级容模型允许的任何地方；Signature 内容模型应出现在 Object 中。本章为可选内容，应用系统可根据实际情况选用。

7.2 Manifest 元素

标识符：

Type=http://127.0.0.1/2000/09/xmldsig#Manifest（可用在 Reference 元素中表示类型）

Manifest 元素提供一张 Reference 列表，与 SignedInfo 中的列表不同，Manifest 列表中的应用程序应规定实际上使用了哪个密码杂凑算法来核对引用的数据对象，还应规定如果对象不可访问或者杂凑值比较失败时，采取什么措施。若从 SignedInfo 中指向 Manifest，应用签名确认行为来验证 Manifest 本身的杂凑值。应用程序可自主决定如何验证 Manifest 中的杂凑值。若从一个 Manifest 中引用另一个 Manifest，则导致不能验证这种两层嵌套的杂凑值。

模式定义：

```
<element name="Manifest" type="ds:ManifestType"/>
<complexType name="ManifestType">
    <sequence>
        <element ref="ds:Reference" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="Id" type="ID" use="optional"/>
</complexType>
```

文档类型定义：

```
<! ELEMENT Manifest (Reference+) >
<! ATTLIST Manifest
                Id ID #IMPLIED >
```

7.3 SignatureProperties 元素

标识符：

Type=http://127.0.0.1/2000/09/xmldsig#SignatureProperties（可用在 Reference 元素中来表示类型。）

SignatureProperty 元素中应存放关于签名生成的任何附加信息（如：时间/日期戳，生成签名时使用的密码硬件的序列号）。

模式定义：

```
<element name="SignatureProperties" type="ds:SignaturePropertiesType"/>
<complexType name="SignaturePropertiesType">
    <sequence>
        <element ref="ds:SignatureProperty" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="Id" type="ID" use="optional"/>
</complexType>

<element name="SignatureProperty" type="ds:SignaturePropertyType"/>
<complexType name="SignaturePropertyType" mixed="true">
    <choice maxOccurs="unbounded">
        <any namespace="##other" processContents="lax"/>
        <! -- (1,1) elements from (1,unbounded) namespaces -->
    </choice>
    <attribute name="Target" type="anyURI" use="required"/>
```

```
<attribute name="Id" type="ID" use="optional"/>
</complexType>
```

文档类型定义：

```
<! ELEMENT SignatureProperties (SignatureProperty+) >
<! ATTLIST SignatureProperties
                Id                    ID    #IMPLIED >
<! ELEMENT SignatureProperty (#PCDATA %SignatureProperty.ANY;)* >
<! ATTLIST SignatureProperty
                Target        CDATA        #REQUIRED
                I             ID           #IMPLIED >
```

7.4 Signature 元素中的处理指令

本标准的语法与处理没有使用 XML 处理指令。

注：如果 CanonicalizationMethod 没有去掉应用程序放在 SignedInfo 中的处理指令，那么它们也会被进行签名处理。所有 CanonicalizationMethod 都包含处理指令。如果一个处理指令是签名处理内容的一部分，那么对处理指令的任何改变都会导致签名失败。

7.5 Signature 元素中的注释

本标准的语法与处理没有使用 XML 注释。

注：如果 CanonicalizationMethod 不去掉 SignedInfo 中的注释或者其他引用的 XML，那么它们都会被进行签名处理。因此，如果保留了它们，任何注释的改变都会导致签名失败；同样，如果不指定忽视注释的规范化/变换方法（如规范化的 XML），对于任何 XML 数据的 XML 签名都会受到注释变化的影响。

8 证实方法

使用人员在检测签名消息是应验证其是否符合 6.1、6.2、6.3、6.4、6.5 及 6.6 的要求，若使用了附加的签名消息，还应验证是否符合 7.2、7.3 的要求，各个元素的具体示例参见附录 A。

附　录　A
（资料性附录）
XML 数字签名实例

A.1　简单实例

A.1.1　概述

下面是符合 XML 规范的描述 HTML4 内容的分离签名的例子。

```
[s01]　　<Signature Id="MyFirstSignature" xmlns="http://127.0.0.1/2000/09/xmldsig#">
[s02]　　<SignedInfo>
[s03]　　<CanonicalizationMethod Algorithm=" http://127.0.0.1/2006/12/xmlc14n11"/>
[s04a]　　<SignatureMethod
[s04b]　　Algorithm="http://127.0.0.1/2001/04/xmldsig-more#sm2-sm3"/>
[s05]　　<Reference URI="http://127.0.0.1/TR/2000/REC-xhtml1—20000126/">
[s06]　　<Transforms>
[s07]　　<Transform Algorithm=" http://127.0.0.1/2006/12/xmlc14n11"/>
[s08]　　</Transforms>
[s09a]　　<DigestMethod
[s09b]　　Algorithm="http://127.0.0.1/2001/04/xmldsig-more#sm3"/>
[s10a]　　<DigestValue> BZT3Gm27GRJs+4fdlf2J+vNWZ5ybI48cg04PtaIJUhE=
[s10b]　　</DigestValue>
[s11]　　</Reference>
[s12]　　</SignedInfo>
[s13a]　　<SignatureValue> x114qrUK7I8fU94Xfmh88DUaLeaSAeWpy6c/
[s13b]　　jbzohF6qszDkvmHy0FDufomhi30asLEkP2DCdfeeM5IMfIpn0w==</SignatureValue>
[s14]　　<KeyInfo>
[s15a]　　<KeyValue>
[s15b]　　<SM2KeyValue xmlns="http://127.0.0.1/2009/xmldsig11#">
[s15c]　　<NamedCurve URI="urn:oid:1.2.156.10197.1.301" />
[s15d]　　<PublicKey>BMKRxUw7j8ThPCi8FG4n4k4xm6XhsgcM0K5LIO86L
[s15e]　　HVqUOl+R3pLeNAVhg00yjXZispILbmKuImCIIrEin8fH68=</PublicKey>
[s15f]　　</SM2KeyValue >
[s15g]　　</KeyValue>
[s16]　　</KeyInfo>
[s17]　　</Signature>
```

其中，[s02-12] SignedInfo 元素是必需的，实际上它就是要进行签名的信息。SingedInfo 的确认包

括两个必备的处理过程：在 SingedInfo 上的 Signature 确认和在 SingedInfo 中的每个引用杂凑的确认。要注意的是，计算 SignatureValue 使用的算法在 SignatureValue 元素处于 SignedInfo 范围外时，也被包括在要签名的信息中。

[s03] 把 SignedInfo 元素当作签名操作的一部分进行杂凑处理之前，先使用规范化算法对它进行规范化。要注意，这个例子以及本附录的所有的例子都不是规范化的形式。

[s04] 使用 SignatureMethod 算法把规范化的 SignedInfo 转换成 SignatureValue，它是一个密码杂凑算法和一个依赖于密钥的算法，还可能和其他算法（如 RSA-SHA1 的填充算法）的结合，将算法名字进行签名，以抵御那种替换成算法复杂度较弱的算法的攻击。为了提高应用互操作性，尽管是否使用这些算法完全取决于签名的创建者，本附录仍列举了一系列的签名算法，本附录推荐了一些，也允许用户声明自己的算法。

[s05-11] 每个 Reference 元素包括密码杂凑算法和对标识出的数据对象进行处理后的杂凑结果，也可包括产生杂凑处理输入的变换，通过计算数据对象的杂凑值并且对该值进行签名，从而为该数据对象加上签名，随后可通过引用确认和签名确认来检验签名结果。

[s14-16] KeyInfo 指示用来确认签名的公钥。标识的可能形式包括证书，密钥名称和密钥交换算法和信息。KeyInfo 是可选的有两个原因，第一：签名者可能不愿意把公钥信息展现给文档处理的各方；第二：公钥信息可能在应用的上下文中就能够得到，不用显式的表示出来。由于 KeyInfo 在 Singed-Info 的外部，如果签名者想把公钥信息也加入签名信息中，使用 Reference 可很容易来标识它并把 Key-Info 加入签名中。

A.1.2 更多 Reference 的内容

```
[s05] <Reference URI="http://127.0.0.1/TR/2000/REC-xhtml1-20000126/">
[s06]     <Transforms>
[s07]         <Transform Algorithm="http://127.0.0.1/2006/12/xml-c14n11"/>
[s08]     </Transforms>
[s09]     <DigestMethod Algorithm="http://127.0.0.1/2001/04/xmldisg-more#sm3"/>
[s10]     <DigestValue>BZT3Gm27GRJs+4fdlf2J+vNWZ5ybI48cg04PtaIJUhE=</Di-
gestValue>
[s11] </Reference>
```

其中，[s05] Reference 可选的 URI 属性标识待签名的数据对象。在一个数字签名中，最多只能有一个 Reference 忽略该属性。

[s05-s08]此标识与变换一起描述了由签名者提供以何种形式进行杂凑处理来获得签名数据信息。只要杂凑核实，验证者可用另外一种方法来获得杂凑值。特殊情况是，验证者有可能从一个不同的位置来获得内容，比如说从本地存储而不是 URI 指定的地方。

[s06-s08] 变换是一个可选的有序的处理步骤列表，在对资源内容进行杂凑处理之前先用这些步骤对它们进行处理。变换可能包括如下的操作：规范化，编码、解码（包括压缩和解压缩），XSLT，Xpath，XML Schema 验证或者 XInclude。Xpath 变换使得签名者获得忽略源文档其中一部分的文档。因此，那些排除在外的文档可进行改变而不会影响签名的有效性。例如，如果被签名的资源包括签名本身，就得使用这种变换把签名从它自身的计算中排除出去。如果变换元素不存在，资源的内容将被直接进行杂凑处理。尽管规范已经推荐了应有的（和可选的）规范化和解码算法，用户指定自己的变换也是允许的。

[s09-s10] 对数据进行变换操作后，使用 DigestMethod 算法来产生 DigestValue。对 DigestValue

的签名就是把资源的内容和签署者的密钥绑定到一起。

A.2 Object 和 SignatureProperty 扩展实例

本附录展示了 XML 数字签名处理的概念(数据完整性和消息认证),希望表示其他语义的应用应依赖诸如 XML, RDF 等其他的技术。签名应用程序应注意它所签名的内容,接收应用程序但不必明白那些语义。任何关于签名生成的内容都可位于 SignatureProperty 元素中。必需的 Target 属性把 Signature 元素指向属性应用到的 Signature 元素上。

考虑上面的例子,加入一个附加的指向一个包括 SignatureProperty 元素的本地对象的引用。

```
[  ]<Signature Id="MySecondSignature" ...>
[p01]  <SignedInfo>
[  ]          ...
[p02]          <Reference URI="http://127.0.0.1/TR/xml-stylesheet/">
[  ]          ...
[p03]          <Reference URI="#AMadeUpTimeStamp"
[p04]     Type="http://127.0.0.1/2000/09/xmldsig#SignatureProperties">
[p05]          <Transforms>
[p06]               <Transform Algorithm="http://127.0.0.1/2006/12/xml-c14n11"/>
[p07]          </Transforms>
[p08]          <DigestMethod Algorithm="http://127.0.0.1/2001/04/xmldisg-more#sm3"/>
[p09]       <DigestValue>BZT3Gm27GRJs+4fdlf2J+vNWZ5ybI48cg04PtaIJUhE=</Di-
gestValue>
[p10]          </Reference>
[p11]  </SignedInfo>
[p12] ...
[p13] <Object>
[p14]          <SignatureProperties>
[p15]          <SignatureProperty Id="AMadeUpTimeStamp" Target="#MySecondSignature">
[p16]               <timestamp xmlns="http://www.ietf.org/rfcXXXX.txt">
[p17]               <date>19990914</date>
[p18]               <time>14:34:34:34</time>
[p19]               </timestamp>
[p20]          </SignatureProperty>
[p21]          </SignatureProperties>
[p22]    </Object>
[p23] </Signature>
```

其中,[p04] 可选 Reference 中的 Type 属性提供了由 URI 标识的资源的相关信息,可指明是 Object、SignatureProperty 还是 Manifest 元素,应用程序可根据这一点来对一些 Reference 元素进行特殊的初始化处理。指向一个 Object 元素中的一个 XML 数据元素的引用应该标明它指向的真正的元素,如果元素内容不是 XML(也许是二进制或者编码后的数据),引用应标明 Object 和 Reference 类型;如果是,应指明 Object。要注意的是,Type 是一个辅助性的属性,签名不需要对它采取任何处理,也不用

检查它的正确性。

[p13] Object 是一个可选的元素，用于在 Signature 元素或者其他地方引入数据对象。可选择对 Object 进行编码。

[p14-s21] 签名属性，例如签名时间，可选地通过在 Reference 里面指明它们而对它们进行签名。

A.3 Object 和 Manifest 扩展实例

本附录使用 Manifest 元素来展示如何满足附加的一些需求。下面是两个需求以及 Manifes 是如何满足它们的：

a) 尽管签名操作本身是开销很大的公开密钥签名，应用程序仍然经常需要高效地对多个数据对象进行签名操作。这个需要可通过在 SingedInfo 里面包含多个 Reference 元素来实现，因为加入的每个杂凑都保证了进行杂凑处理的数据的安全。然而，一些应用也许不需要这种方法伴随而来的确认行为，因为它需要 SingedInfo 中的每个 Reference 都经过引用确认过程的处理，这些应用可能希望为它们保留引用确认决策逻辑。例如，一个应用可能收到一个包含三个 Reference 元素的 Signature 的合法的 SingedInfo 元素。如果其中一个 Reference 操作失败了，签名就会在确认过程中失败。然而，应用程序可能就想把两个有效的 Reference 元素当作有效来处理，或者就想根据不同的失败原因采取不同的对策，为了实现这一点，SingedInfo 要参考包含一个或多个 Reference 元素的 Manifest 元素。然后，Manifest 的引用确认就处于应用程序的控制中了。

b) 考虑这样一种应用，对大量的文档进行了许多签名操作(使用不同的密钥)。一个效率不太高的解决方案是使用一个分离的签名反复地应用到大量的 SingedInfo 元素中(使用很多 Reference)；这种方法很浪费而且很冗余。一个高效的方法是在一个 Manifest 元素中加入多个 Reference，然后在多个 Signature 元素中引用这个 Manifest。

下面的例子包括了一个 Reference，它签署了一个在 Object 元素中的 Manifest。

```
[ ]...
[m01] <Reference URI="#MyFirstManifest"
[m02]   Type="http://127.0.0.1/2000/09/xmldsig#Manifest">
[m03]     <Transforms>
[m04]       <Transform Algorithm="http://127.0.0.1/2006/12/xml-c14n11"/>
[m05]     </Transforms>
[m06]     <DigestMethod Algorithm="http://127.0.0.1/2001/04/xmldsig-more#sm3"/>
[m07]      <DigestValue>8/LEOkflljEE/fFSEfFTptZrLmUh00SAJUhH3BIPbx0=</Di-
gestValue>
    [m08] </Reference>
    [ ]...
    [m09] <Object>
    [m10] <Manifest Id="MyFirstManifest">
    [m11]   <Reference>
    [m12]   ...
    [m13]   </Reference>
    [m14]   <Reference>
```

[m15] ...
[m16] </Reference>
[m17] </Manifest>
[m18] </Object>

A.4 签名实例

A.4.1 分离签名

签名结果：

```
<? xml version="1.0" encoding="UTF-8" standalone="no"? >
<Signature xmlns="http://127.0.0.1/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod
        Algorithm="http://127.0.0.1/2006/12/xml-c14n11#WithComments"/>
    <SignatureMethod Algorithm="http://127.0.0.1/2001/04/xmldsig-more#sm2-sm3"/>
    <Reference URI="http://127.0.0.1/TR/xml-stylesheet">
      <DigestMethod Algorithm="http://127.0.0.1/2001/04/xmldsig-more#sm3"/>
      <DigestValue>BZT3Gm27GRJs+4fdlf2J+vNWZ5ybI48cg04PtaIJUhE=</DigestValue>
    </Reference>
  </SignedInfo>
                    <SignatureValue>
x114qrUK7I8fU94Xfmh88DUaLeaSAeWpy6c/jbzohF6qszDkvmHy0FDufomhi30asLEkP2DCdfee
M5IMfIpn0w==</SignatureValue>
  <KeyInfo>
    <KeyValue>
      <ECKeyValue xmlns="http://127.0.0.1/2009/xmldsig11#">
        <NamedCurve URI="urn:oid:1.2.840.10045.3.1.7"/>
<PublicKey>
BMKRxUw7j8ThPCi8FG4n4k4xm6XhsgcM0K5LIO86LHVqUOl+R3pLeNAVhg00yjXZispI Lb-
mKuImCIIrEin8fH68=</PublicKey>
      </ECKeyValue>
    </KeyValue>
  </KeyInfo>
</Signature>
```

A.4.2 封内签名

签名后：

```
<? xml version="1.0" encoding="UTF-8" standalone="no"? >
<Signature xmlns="http://127.0.0.1/2000/09/xmldsig#">
<SignedInfo>
<CanonicalizationMethod
```

```
    Algorithm="http://127.0.0.1/2006/12/xml-c14n11#WithComments"/>
          <SignatureMethod Algorithm="http://127.0.0.1/2001/04/xmldsig-more#sm2-
sm3"/>
        <Reference URI="#object">
          <DigestMethod Algorithm="http://127.0.0.1/2001/04/xmldsig-more#sm3"/>
          <DigestValue>yG0h7G1g/ngi1B5NE+UsSQmtz9TnZUqhr08B0KJGr3E=</Di-
gestValue>
        </Reference>
    </SignedInfo>
    <SignatureValue>
    zhHGxS4xCV1D6R2oM7GmDM2Kh8m + sWntGuDu33gVENr2nC2YUqBDPi7b9t707x4ijq6sA6
Re9yf9fDuPYEPsJg==</SignatureValue>
        <KeyInfo>
          <KeyValue>
            <ECKeyValue xmlns="http://127.0.0.1/2009/xmldsig11#">
              <NamedCurve URI="urn:oid:1.2.840.10045.3.1.7"/>
              <PublicKey>
                BNPobUqvBfYbo9zNDZvYrZiXGME2ACNDQnco1fooRNzrPMIZlSCko
                AZ3jgwdeVpM9A9c6AW5MEMk +I/gOqZ/woU=</PublicKey>
            </ECKeyValue>
          </KeyValue>
        </KeyInfo>
        <Object Id="object">some text</Object>
    </Signature>
```

A.4.3 封皮签名

签名前：

```
<Envelope xmlns="urn:envelope">
</Envelope>
```

签名后：

```
<? xml version="1.0" encoding="UTF-8"? >
<Envelope xmlns="urn:envelope">
    <Signature xmlns="http://127.0.0.1/2000/09/xmldsig#">
        <SignedInfo>
            <CanonicalizationMethod
            Algorithm="http://127.0.0.1/2006/12/xml-c14n11#WithComments"/>
              <SignatureMethod Algorithm="http://127.0.0.1/2001/04/xmldsig-more#sm2-
sm3"/>
            <Reference URI="">
              <Transforms>
                <Transform
```

```
                Algorithm="http://127.0.0.1/2000/09/xmldsig#enveloped-signature"/>
                </Transforms>
                <DigestMethod Algorithm="http://127.0.0.1/2001/04/xmldsig-more#sm3"/>
                 <DigestValue> hLA10BfAKncPgRR7cCD8wlm/s9Fr/Wm85EKzOdy4dIg=</
DigestValue>
                </Reference>
            </SignedInfo>
            <SignatureValue>
        DHZ7uQSzGLkJ2DSZxtGbXMT10yoCUq9wryTTMZn  9FFwEcEDO0o3y6zowm4O7lHr91m
As45kSiMt8WurE0qEsCQ==</SignatureValue>
            <KeyInfo>
                <KeyValue>
                    <ECKeyValue xmlns="http://127.0.0.1/2009/xmldsig11#">
                        <NamedCurve URI="urn:oid:1.2.840.10045.3.1.7"/>
                        <PublicKey>
    BGr10WM5ieRvi1RhHK/WLbUKgd + rQ3PHtHj1nmJ/lHpobsZZuT6qaGMYs/IF4j9HilRLx
6ttP9xGENbq/6L5LYc=</PublicKey>
                    </ECKeyValue>
                </KeyValue>
            </KeyInfo>
        </Signature>
    </Envelope>
```

附 录 B
（规范性附录）
XML 数字签名文档类型定义

```
<! -- DTD for XML Schemas: Part 1: Structures
     Public Identifier: "-//W3C//DTD XMLSCHEMA 200102//EN"
     Official Location: http://127.0.0.1/2001/XMLSchema.dtd -->
<! -- $Id: XMLSchema.dtd,v 1.31 2001/10/24 15:50:16 ht Exp $ -->
<! -- Note this DTD is NOT normative, or even definitive. -->                    <! --d-->
<! -- prose copy in the structures REC is the definitive version -->             <! --d-->
<! -- (which shouldn't differ from this one except for this -->                  <! --d-->
<! -- comment and entity expansions, but just in case) -->                       <! --d-->
<! -- With the exception of cases with multiple namespace
     prefixes for the XML Schema namespace, any XML document which is
     not valid per this DTD given redefinitions in its internal subset of the
     'p' and 's' parameter entities below appropriate to its namespace
     declaration of the XML Schema namespace is almost certainly not
     a valid schema. -->

<! -- The simpleType element and its constituent parts
     are defined in XML Schema: Part 2: Datatypes -->
<! ENTITY % xs-datatypes PUBLIC 'datatypes' 'datatypes.dtd' >

<! ENTITY % p 'xs:'> <! -- can be overriden in the internal subset of a
                          schema document to establish a different
                          namespace prefix -->
<! ENTITY % s ':xs'> <! -- if %p is defined (e.g. as foo:) then you must
                          also define %s as the suffix for the appropriate
                          namespace declaration (e.g. :foo) -->
<! ENTITY % nds 'xmlns%s;'>

<! -- Define all the element names, with optional prefix -->
<! ENTITY % schema "%p;schema">
<! ENTITY % complexType "%p;complexType">
<! ENTITY % complexContent "%p;complexContent">
<! ENTITY % simpleContent "%p;simpleContent">
<! ENTITY % extension "%p;extension">
<! ENTITY % element "%p;element">
<! ENTITY % unique "%p;unique">
<! ENTITY % key "%p;key">
```

```
<! ENTITY % keyref "%p;keyref">
<! ENTITY % selector "%p;selector">
<! ENTITY % field "%p;field">
<! ENTITY % group "%p;group">
<! ENTITY % all "%p;all">
<! ENTITY % choice "%p;choice">
<! ENTITY % sequence "%p;sequence">
<! ENTITY % any "%p;any">
<! ENTITY % anyAttribute "%p;anyAttribute">
<! ENTITY % attribute "%p;attribute">
<! ENTITY % attributeGroup "%p;attributeGroup">
<! ENTITY % include "%p;include">
<! ENTITY % import "%p;import">
<! ENTITY % redefine "%p;redefine">
<! ENTITY % notation "%p;notation">

<! -- annotation elements -->
<! ENTITY % annotation "%p;annotation">
<! ENTITY % appinfo "%p;appinfo">
<! ENTITY % documentation "%p;documentation">

<! -- Customisation entities for the ATTLIST of each element type.
     Define one of these if your schema takes advantage of the
     anyAttribute='##other' in the schema for schemas -->

<! ENTITY % schemaAttrs ''>
<! ENTITY % complexTypeAttrs ''>
<! ENTITY % complexContentAttrs ''>
<! ENTITY % simpleContentAttrs ''>
<! ENTITY % extensionAttrs ''>
<! ENTITY % elementAttrs ''>
<! ENTITY % groupAttrs ''>
<! ENTITY % allAttrs ''>
<! ENTITY % choiceAttrs ''>
<! ENTITY % sequenceAttrs ''>
<! ENTITY % anyAttrs ''>
<! ENTITY % anyAttributeAttrs ''>
<! ENTITY % attributeAttrs ''>
<! ENTITY % attributeGroupAttrs ''>
<! ENTITY % uniqueAttrs ''>
<! ENTITY % keyAttrs ''>
```

```
<! ENTITY % keyrefAttrs ''>
<! ENTITY % selectorAttrs ''>
<! ENTITY % fieldAttrs ''>
<! ENTITY % includeAttrs ''>
<! ENTITY % importAttrs ''>
<! ENTITY % redefineAttrs ''>
<! ENTITY % notationAttrs ''>
<! ENTITY % annotationAttrs ''>
<! ENTITY % appinfoAttrs ''>
<! ENTITY % documentationAttrs ''>

<! ENTITY % complexDerivationSet "CDATA">
    <! -- #all or space-separated list drawn from derivationChoice -->
<! ENTITY % blockSet "CDATA">
    <! -- #all or space-separated list drawn from
                derivationChoice + 'substitution' -->

<! ENTITY % mgs '%all; | %choice; | %sequence;'>
<! ENTITY % cs '%choice; | %sequence;'>
<! ENTITY % formValues '(qualified|unqualified)'>

<! ENTITY % attrDecls '((%attribute;| %attributeGroup;) * ,(%anyAttribute;)?)'>

<! ENTITY % particleAndAttrs '((%mgs; | %group;)?, %attrDecls;)'>

<! -- This is used in part2 -->
<! ENTITY % restriction1 '((%mgs; | %group;)?)'>

%xs-datatypes;

<! -- the duplication below is to produce an unambiguous content model
    which allows annotation everywhere -->
<! ELEMENT %schema; ((%include; | %import; | %redefine; | %annotation;) * ,
                    ((%simpleType; | %complexType;
                    | %element; | %attribute;
                    | %attributeGroup; | %group;
                    | %notation; ),
                    (%annotation;) * ) * )>
<! ATTLIST %schema;
  targetNamespace       %URIref;                #IMPLIED
```

```
        version                 CDATA                       #IMPLIED
        %nds;                   %URIref;    #FIXED 'http://127.0.0.1/2001/XMLSchema'
        xmlns                   CDATA                       #IMPLIED
        finalDefault            %complexDerivationSet;      ''
        blockDefault            %blockSet;                  ''
        id                      ID                          #IMPLIED
        elementFormDefault      %formValues;                'unqualified'
        attributeFormDefault    %formValues;                'unqualified'
        xml:lang                CDATA                       #IMPLIED
        %schemaAttrs;>
<! -- Note the xmlns declaration is NOT in the Schema for Schemas,
        because at the Infoset level where schemas operate,
        xmlns(:prefix) is NOT an attribute! -->
<! -- The declaration of xmlns is a convenience for schema authors -->

<! -- The id attribute here and below is for use in external references
        from non-schemas using simple fragment identifiers.
        It is NOT used for schema-to-schema reference, internal or
        external. -->

<! -- a type is a named content type specification which allows attribute
        declarations-->
<! -- -->

<! ELEMENT %complexType; ((%annotation;)?,
                         (%simpleContent;|%complexContent;|
                         %particleAndAttrs;))>

<! ATTLIST %complexType;
            name        %NCName;                    #IMPLIED
            id          ID                          #IMPLIED
            abstract    %boolean;                   #IMPLIED
            final       %complexDerivationSet;      #IMPLIED
            block       %complexDerivationSet;      #IMPLIED
            mixed (true|false) 'false'
            %complexTypeAttrs;>

  <! -- particleAndAttrs is shorthand for a root type -->
  <! -- mixed is disallowed if simpleContent, overriden if complexContent
          has one too. -->
```

```
<! -- If anyAttribute appears in one or more referenced attributeGroups
        and/or explicitly, the intersection of the permissions is used -->

<! ELEMENT %complexContent; ((%annotation;)?, (%restriction;|%extension;))>
<! ATTLIST %complexContent;
                mixed (true|false) #IMPLIED
                id     ID          #IMPLIED
                %complexContentAttrs;>

<! -- restriction should use the branch defined above, not the simple
        one from part2; extension should use the full model -->

<! ELEMENT %simpleContent; ((%annotation;)?, (%restriction;|%extension;))>
<! ATTLIST %simpleContent;
                id     ID          #IMPLIED
                %simpleContentAttrs;>

<! -- restriction should use the simple branch from part2, not the
        one defined above; extension should have no particle -->

<! ELEMENT %extension; ((%annotation;)?, (%particleAndAttrs;))>
<! ATTLIST %extension;
                base   %QName;     #REQUIRED
                id     ID          #IMPLIED
                %extensionAttrs;>

<! -- an element is declared by either:
   a name and a type (either nested or referenced via the type attribute)
   or a ref to an existing element declaration -->

<! ELEMENT %element; ((%annotation;)?, (%complexType;| %simpleType;)?,
                        (%unique; | %key; | %keyref;)*)>
<! -- simpleType or complexType only if no type|ref attribute -->
<! -- ref not allowed at top level -->
<! ATTLIST %element;
                name        %NCName;            #IMPLIED
                id          ID                  #IMPLIED
                ref         %QName;             #IMPLIED
                type        %QName;             #IMPLIED
                minOccurs   %nonNegativeInteger; #IMPLIED
                maxOccurs   CDATA               #IMPLIED
```

```
                nillable          %boolean;              #IMPLIED
                substitutionGroup %QName;                #IMPLIED
                abstract          %boolean;              #IMPLIED
                final             %complexDerivationSet; #IMPLIED
                block             %blockSet;             #IMPLIED
                default           CDATA                  #IMPLIED
                fixed             CDATA                  #IMPLIED
                form              %formValues;           #IMPLIED
                %elementAttrs;>
<!-- type and ref are mutually exclusive.
     name and ref are mutually exclusive, one is required -->
<!-- In the absence of type AND ref, type defaults to type of
     substitutionGroup, if any, else the ur-type, i.e. unconstrained -->
<!-- default and fixed are mutually exclusive -->

<!ELEMENT %group; ((%annotation;)?,(%mgs;)?)>
<!ATTLIST %group;
                name          %NCName;               #IMPLIED
                ref           %QName;                #IMPLIED
                minOccurs     %nonNegativeInteger;   #IMPLIED
                maxOccurs     CDATA                  #IMPLIED
                id            ID                     #IMPLIED
                %groupAttrs;>

<!ELEMENT %all; ((%annotation;)?, (%element;)*)>
<!ATTLIST %all;
                minOccurs     (1)                    #IMPLIED
                maxOccurs     (1)                    #IMPLIED
                id            ID                     #IMPLIED
                %allAttrs;>

<!ELEMENT %choice; ((%annotation;)?, (%element;| %group;| %cs; | %any;)*)>
<!ATTLIST %choice;
                minOccurs     %nonNegativeInteger;   #IMPLIED
                maxOccurs     CDATA                  #IMPLIED
                id            ID                     #IMPLIED
                %choiceAttrs;>

<!ELEMENT %sequence; ((%annotation;)?, (%element;| %group;| %cs; | %any;)*)>
<!ATTLIST %sequence;
                minOccurs     %nonNegativeInteger;   #IMPLIED
```

```
                maxOccurs    CDATA                    #IMPLIED
                id           ID                       #IMPLIED
                %sequenceAttrs;>

<! -- an anonymous grouping in a model, or
        a top-level named group definition, or a reference to same -->

<! -- Note that if order is 'all', group is not allowed inside.
        If order is 'all' THIS group must be alone (or referenced alone) at
        the top level of a content model -->
<! -- If order is 'all', minOccurs==maxOccurs==1 on element/any inside -->
<! -- Should allow minOccurs=0 inside order='all' . . . -->

<! ELEMENT %any; (%annotation;)? >
<! ATTLIST %any;
                namespace        CDATA                  '##any'
                processContents  (skip|lax|strict)      'strict'
                minOccurs        %nonNegativeInteger;   '1'
                maxOccurs        CDATA                  '1'
                id               ID                     #IMPLIED
                %anyAttrs;>

<! -- namespace is interpreted as follows:
                ##any       -- any non-conflicting WFXML at all

                ##other     -- any non-conflicting WFXML from namespace other
                               than targetNamespace

                ##local     -- any unqualified non-conflicting WFXML/attribute
                one or      -- any non-conflicting WFXML from
                more URI       the listed namespaces
                references

                ##targetNamespace ##local may appear in the above list,
                  with the obvious meaning -->

<! ELEMENT %anyAttribute; (%annotation;)? >
<! ATTLIST %anyAttribute;
                namespace        CDATA                  '##any'
                processContents  (skip|lax|strict)      'strict'
                id               ID                     #IMPLIED
```

```
                %anyAttributeAttrs;>
  <! -- namespace is interpreted as for 'any' above -->

  <! -- simpleType only if no type|ref attribute -->
  <! -- ref not allowed at top level, name iff at top level -->
  <! ELEMENT %attribute; ((%annotation;)?, (%simpleType;)?)>
  <! ATTLIST %attribute;
      name        %NCName;          #IMPLIED
      id          ID                #IMPLIED
      ref         %QName;           #IMPLIED
      type        %QName;           #IMPLIED
      use         (prohibited|optional|required) #IMPLIED
      default     CDATA             #IMPLIED
      fixed       CDATA             #IMPLIED
      form        %formValues;      #IMPLIED
      %attributeAttrs;>
  <! -- type and ref are mutually exclusive.
       name and ref are mutually exclusive, one is required -->
  <! -- default for use is optional when nested, none otherwise -->
  <! -- default and fixed are mutually exclusive -->
  <! -- type attr and simpleType content are mutually exclusive -->

  <! -- an attributeGroup is a named collection of attribute decls, or a
       reference thereto -->
  <! ELEMENT %attributeGroup; ((%annotation;)?,
                    (%attribute; | %attributeGroup;)*,
                    (%anyAttribute;)?) >
  <! ATTLIST %attributeGroup;
      name        %NCName;          #IMPLIED
      id          ID                #IMPLIED
      ref         %QName;           #IMPLIED
      %attributeGroupAttrs;>

  <! -- ref iff no content, no name. ref iff not top level -->

  <! -- better reference mechanisms -->
  <! ELEMENT %unique; ((%annotation;)?, %selector;, (%field;)+)>
  <! ATTLIST %unique;
      name        %NCName;          #REQUIRED
      id          ID                #IMPLIED
      %uniqueAttrs;>
```

```
<! ELEMENT %key; ((%annotation;)?, %selector;, (%field;)+)>
<! ATTLIST %key;
                    name        %NCName;          #REQUIRED
                    id          ID                #IMPLIED
                    %keyAttrs;>

<! ELEMENT %keyref; ((%annotation;)?, %selector;, (%field;)+)>
<! ATTLIST %keyref;
                        name          %NCName;          #REQUIRED
                                      refer   %QName;   #REQUIRED
                                          id      ID    #IMPLIED
                                              %keyrefAttrs;>

<! ELEMENT %selector; ((%annotation;)?)>
<! ATTLIST %selector;
                xpath %XPathExpr; #REQUIRED
                id        ID          #IMPLIED
                %selectorAttrs;>
<! ELEMENT %field; ((%annotation;)?)>
<! ATTLIST %field;
                xpath %XPathExpr; #REQUIRED
                id        ID          #IMPLIED
                %fieldAttrs;>

<! -- Schema combination mechanisms -->
<! ELEMENT %include; (%annotation;)? >
<! ATTLIST %include;
                schemaLocation %URIref; #REQUIRED
                id                  ID        #IMPLIED
                %includeAttrs;>

<! ELEMENT %import; (%annotation;)? >
<! ATTLIST %import;
                namespace         %URIref; #IMPLIED
                schemaLocation    %URIref; #IMPLIED
                id                ID       #IMPLIED
                %importAttrs;>

<! ELEMENT %redefine; (%annotation; | %simpleType; | %complexType; |
                        %attributeGroup; | %group;)*>
<! ATTLIST %redefine;
```

```
                schemaLocation %URIref; #REQUIRED
                id              ID      #IMPLIED
                %redefineAttrs;>

<! ELEMENT %notation; (%annotation;)? >
<! ATTLIST %notation;
                        name        %NCName;        #REQUIRED
                        id          ID              #IMPLIED
                        public      CDATA           #REQUIRED
                        system      %URIref;        #IMPLIED
                        %notationAttrs;>

<! -- Annotation is either application information or documentation -->
<! -- By having these here they are available for datatypes as well
     as all the structures elements -->

<! ELEMENT %annotation; (%appinfo; | %documentation;)* >
<! ATTLIST %annotation; %annotationAttrs;>

<! -- User must define annotation elements in internal subset for this
     to work -->
<! ELEMENT %appinfo; ANY>  <! -- too restrictive -->
<! ATTLIST %appinfo;
                source      %URIref;        #IMPLIED
                id          ID              #IMPLIED
                %appinfoAttrs;>
<! ELEMENT %documentation; ANY>  <! -- too restrictive -->
<! ATTLIST %documentation;
                source      %URIref;        #IMPLIED
                id          ID              #IMPLIED
                xml:lang    CDATA           #IMPLIED
                %documentationAttrs;>

<! NOTATION XMLSchemaStructures PUBLIC
                'structures' 'http://127.0.0.1/2001/XMLSchema.xsd' >
<! NOTATION XML PUBLIC
                'REC-xml-1998-0210' 'http://127.0.0.1/TR/1998/REC-xml-19980210' >
```

附　录　C
（规范性附录）
XML 数字签名模式定义

C.1　xmldisg-core-schema.xsd

```
<? xml version="1.0" encoding="UTF-8"? >
<! DOCTYPE schema PUBLIC "-//W3C//DTD XMLSchema 200102//EN" "http://127.0.0.
1/2001/XMLSchema.dtd" [
    <! ATTLIST schema
      xmlns:ds CDATA #FIXED "http://127.0.0.1/2000/09/xmldsig#">
    <! ENTITY dsig 'http://127.0.0.1/2000/09/xmldsig#'>
    <! ENTITY % p ''>
    <! ENTITY % s ''>
   ]>
<! -- Schema for XML Signatures
     http://127.0.0.1/2000/09/xmldsig#
     $Revision: 1.1 $ on $Date: 2002/02/08 20:32:26 $ by $Author: reagle$-->
    <schema xmlns="http://127.0.0.1/2001/XMLSchema"
xmlns:ds="http://127.0.0.1/2000/09/xmldsig#"
targetNamespace="http://127.0.0.1/2000/09/xmldsig#" version="0.1"
elementFormDefault="qualified">

    <! -- Basic Types Defined for Signatures -->

    <simpleType name="CryptoBinary">
      <restriction base="base64Binary">
      </restriction>
    </simpleType>

    <! -- Start Signature -->

    <element name="Signature" type="ds:SignatureType"/>
    <complexType name="SignatureType">
      <sequence>
        <element ref="ds:SignedInfo"/>
        <element ref="ds:SignatureValue"/>
        <element ref="ds:KeyInfo" minOccurs="0"/>
        <element ref="ds:Object" minOccurs="0" maxOccurs="unbounded"/>
```

```
    </sequence>
    <attribute name="Id" type="ID" use="optional"/>
  </complexType>

    <element name="SignatureValue" type="ds:SignatureValueType"/>
    <complexType name="SignatureValueType">
      <simpleContent>
        <extension base="base64Binary">
          <attribute name="Id" type="ID" use="optional"/>
        </extension>
      </simpleContent>
    </complexType>

<! -- Start SignedInfo -->

<element name="SignedInfo" type="ds:SignedInfoType"/>
<complexType name="SignedInfoType">
  <sequence>
    <element ref="ds:CanonicalizationMethod"/>
    <element ref="ds:SignatureMethod"/>
    <element ref="ds:Reference" maxOccurs="unbounded"/>
  </sequence>
  <attribute name="Id" type="ID" use="optional"/>
</complexType>

  <element name="CanonicalizationMethod" type="ds:CanonicalizationMethodType"/>
  <complexType name="CanonicalizationMethodType" mixed="true">
    <sequence>
      <any namespace="##any" minOccurs="0" maxOccurs="unbounded"/>
      <! -- (0,unbounded) elements from (1,1) namespace -->
    </sequence>
    <attribute name="Algorithm" type="anyURI" use="required"/>
  </complexType>

    <element name="SignatureMethod" type="ds:SignatureMethodType"/>
    <complexType name="SignatureMethodType" mixed="true">
      <sequence>
        <element name="HMACOutputLength" minOccurs="0" type="ds:HMACOutputLe-
ngthType"/>
        <any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
        <! -- (0,unbounded) elements from (1,1) external namespace -->
```

```
    </sequence>
    <attribute name="Algorithm" type="anyURI" use="required"/>
  </complexType>

<! -- Start Reference -->

<element name="Reference" type="ds:ReferenceType"/>
<complexType name="ReferenceType">
  <sequence>
    <element ref="ds:Transforms" minOccurs="0"/>
    <element ref="ds:DigestMethod"/>
    <element ref="ds:DigestValue"/>
  </sequence>
  <attribute name="Id" type="ID" use="optional"/>
  <attribute name="URI" type="anyURI" use="optional"/>
  <attribute name="Type" type="anyURI" use="optional"/>
</complexType>

  <element name="Transforms" type="ds:TransformsType"/>
  <complexType name="TransformsType">
    <sequence>
      <element ref="ds:Transform" maxOccurs="unbounded"/>
    </sequence>
</complexType>

<element name="Transform" type="ds:TransformType"/>
<complexType name="TransformType" mixed="true">
  <choice minOccurs="0" maxOccurs="unbounded">
    <any namespace="##other" processContents="lax"/>
    <! -- (1,1) elements from (0,unbounded) namespaces -->
    <element name="XPath" type="string"/>
  </choice>
    <attribute name="Algorithm" type="anyURI" use="required"/>
  </complexType>

<! -- End Reference -->

<element name="DigestMethod" type="ds:DigestMethodType"/>
<complexType name="DigestMethodType" mixed="true">
  <sequence>
    <any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="un-
```

```
bounded"/>
        </sequence>
        <attribute name="Algorithm" type="anyURI" use="required"/>
      </complexType>

      <element name="DigestValue" type="ds:DigestValueType"/>
      <simpleType name="DigestValueType">
        <restriction base="base64Binary"/>
      </simpleType>

      <! -- End SignedInfo -->

      <! -- Start KeyInfo -->

      <element name="KeyInfo" type="ds:KeyInfoType"/>
      <complexType name="KeyInfoType" mixed="true">
        <choice maxOccurs="unbounded">
          <element ref="ds:KeyName"/>
          <element ref="ds:KeyValue"/>
          <element ref="ds:RetrievalMethod"/>
          <element ref="ds:X509Data"/>
          <any processContents="lax" namespace="##other"/>
          <! -- (1,1) elements from (0,unbounded) namespaces -->
        </choice>
        <attribute name="Id" type="ID" use="optional"/>
      </complexType>

        <element name="KeyName" type="string"/>
        <element name="MgmtData" type="string"/>

        <element name="KeyValue" type="ds:KeyValueType"/>
        <complexType name="KeyValueType" mixed="true">
          <choice>
            <element ref="ds:RSAKeyValue"/>
            <any namespace="##other" processContents="lax"/>
          </choice>
        </complexType>

        <element name="RetrievalMethod" type="ds:RetrievalMethodType"/>
        <complexType name="RetrievalMethodType">
          <sequence>
```

```
      <element ref="ds:Transforms" minOccurs="0"/>
    </sequence>
    <attribute name="URI" type="anyURI"/>
    <attribute name="Type" type="anyURI" use="optional"/>
  </complexType>

<! -- Start X509Data -->

<element name="X509Data" type="ds:X509DataType"/>
<complexType name="X509DataType">
  <sequence maxOccurs="unbounded">
    <choice>
      <element name="X509IssuerSerial" type="ds:X509IssuerSerialType"/>
      <element name="X509SKI" type="base64Binary"/>
      <element name="X509SubjectName" type="string"/>
      <element name="X509Certificate" type="base64Binary"/>
      <element name="X509CRL" type="base64Binary"/>
      <any namespace="##other" processContents="lax"/>
    </choice>
  </sequence>
</complexType>

<complexType name="X509IssuerSerialType">
  <sequence>
    <element name="X509IssuerName" type="string"/>
    <element name="X509SerialNumber" type="integer"/>
  </sequence>
</complexType>

<! -- End X509Data -->

<! -- End KeyInfo -->

<! -- Start Object (Manifest, SignatureProperty) -->

<element name="Object" type="ds:ObjectType"/>
<complexType name="ObjectType" mixed="true">
  <sequence minOccurs="0" maxOccurs="unbounded">
    <any namespace="##any" processContents="lax"/>
  </sequence>
  <attribute name="Id" type="ID" use="optional"/>
```

```
      <attribute name="MimeType" type="string" use="optional"/> <! -- add a grep facet
-->
      <attribute name="Encoding" type="anyURI" use="optional"/>
    </complexType>

    <element name="Manifest" type="ds:ManifestType"/>
    <complexType name="ManifestType">
      <sequence>
        <element ref="ds:Reference" maxOccurs="unbounded"/>
      </sequence>
      <attribute name="Id" type="ID" use="optional"/>
    </complexType>

  <element name="SignatureProperties" type="ds:SignaturePropertiesType"/>
  <complexType name="SignaturePropertiesType">
    <sequence>
      <element ref="ds:SignatureProperty" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="Id" type="ID" use="optional"/>
  </complexType>

    <element name="SignatureProperty" type="ds:SignaturePropertyType"/>
    <complexType name="SignaturePropertyType" mixed="true">
      <choice maxOccurs="unbounded">
        <any namespace="##other" processContents="lax"/>
        <! -- (1,1) elements from (1,unbounded) namespaces -->
      </choice>
      <attribute name="Target" type="anyURI" use="required"/>
      <attribute name="Id" type="ID" use="optional"/>
    </complexType>

  <! -- End Object (Manifest, SignatureProperty) -->

    <! -- Start Algorithm Parameters -->

    <simpleType name="HMACOutputLengthType">
      <restriction base="integer"/>
    </simpleType>

    <! -- Start KeyValue Element-types -->
```

```
<element name="RSAKeyValue" type="ds:RSAKeyValueType"/>
<complexType name="RSAKeyValueType">
  <sequence>
    <element name="Modulus" type="ds:CryptoBinary"/>
    <element name="Exponent" type="ds:CryptoBinary"/>
  </sequence>
</complexType>

    <! -- End KeyValue Element-types -->

    <! -- End Signature -->

    </schema>
```

C.2 xmldsig11-schema.xsd

```
    <? xml version="1.0" encoding="utf-8"? >

    <schema xmlns="http://127.0.0.1/2001/XMLSchema"
            xmlns:ds="http://127.0.0.1/2000/09/xmldsig#"
            xmlns:dsig11="http://127.0.0.1/2009/xmldsig11#"
            targetNamespace="http://127.0.0.1/2009/xmldsig11#"
            version="0.1" elementFormDefault="qualified">

  <import namespace=http://127.0.0.1/2000/09/xmldsig# />

<! --Begin SM2KeyValue -->
  <element name="SM2KeyValue" type="dsig11:SM2KeyValueType"/>
  <complexType name="SM2KeyValueType">
    <sequence>
        <element name="NamedCurve" type="dsig11:NamedCurveType"/>
      <element name="PublicKey" type="dsig11:ECPointType"/>
    </sequence>
    <attribute name="Id" type="ID" use="optional"/>
  </complexType>
<! --End SM2KeyValue -->

  <element name="ECKeyValue" type="dsig11:ECKeyValueType"/>
  <complexType name="ECKeyValueType">
    <sequence>
      <choice>
```

```
        <element name="ECParameters" type="dsig11:ECParametersType"/>
        <element name="NamedCurve" type="dsig11:NamedCurveType"/>
      </choice>
      <element name="PublicKey" type="dsig11:ECPointType"/>
    </sequence>
    <attribute name="Id" type="ID" use="optional"/>
  </complexType>

  <complexType name="NamedCurveType">
    <attribute name="URI" type="anyURI" use="required"/>
  </complexType>

  <simpleType name="ECPointType">
    <restriction base="ds:CryptoBinary"/>
  </simpleType>

  <complexType name="ECParametersType">
    <sequence>
      <element name="FieldID" type="dsig11:FieldIDType"/>
      <element name="Curve" type="dsig11:CurveType"/>
      <element name="Base" type="dsig11:ECPointType"/>
      <element name="Order" type="ds:CryptoBinary"/>
      <element name="CoFactor" type="integer" minOccurs="0"/>
      <element name="ValidationData"
               type="dsig11:ECValidationDataType" minOccurs="0"/>
    </sequence>
  </complexType>

  <complexType name="FieldIDType">
    <choice>
      <element ref="dsig11:Prime"/>
      <any namespace="##other" processContents="lax"/>
    </choice>
  </complexType>

  <complexType name="CurveType">
    <sequence>
      <element name="A" type="ds:CryptoBinary"/>
      <element name="B" type="ds:CryptoBinary"/>
    </sequence>
  </complexType>
```

```
<complexType name="ECValidationDataType">
  <sequence>
    <element name="seed" type="ds:CryptoBinary"/>
  </sequence>
  <attribute name="hashAlgorithm" type="anyURI" use="required"/>
</complexType>

<element name="Prime" type="dsig11:PrimeFieldParamsType"/>
<complexType name="PrimeFieldParamsType">
  <sequence>
    <element name="P" type="ds:CryptoBinary"/>
  </sequence>
</complexType>

<element name="DEREncodedKeyValue" type="dsig11:DEREncodedKeyValueType"/>
<complexType name="DEREncodedKeyValueType">
  <simpleContent>
    <extension base="base64Binary">
      <attribute name="Id" type="ID" use="optional"/>
    </extension>
  </simpleContent>
</complexType>

<element name="KeyInfoReference" type="dsig11:KeyInfoReferenceType"/>
<complexType name="KeyInfoReferenceType">
  <attribute name="URI" type="anyURI" use="required"/>
  <attribute name="Id" type="ID" use="optional"/>
</complexType>

<element name="X509Digest" type="dsig11:X509DigestType"/>
<complexType name="X509DigestType">
  <simpleContent>
    <extension base="base64Binary">
      <attribute name="Algorithm" type="anyURI" use="required"/>
    </extension>
  </simpleContent>
</complexType>

</schema>
```

C.3 xmldsig1-schema.xsd

```
<? xml version="1.0" encoding="utf-8"? >

<schema xmlns="http://127.0.0.1/2001/XMLSchema"
                    targetNamespace="http://127.0.0.1/2000/09/xmldsig#"
                    version="0.1" elementFormDefault="qualified">

    <include
  schemaLocation="http://127.0.0.1/TR/2008/REC-xmldsig-core-20080610/xmldsig-core-
schema.xsd"/>

    <import namespace="http://127.0.0.1/2009/xmldsig11#"
     schemaLocation="http://127.0.0.1/TR/xmldsig-core1/xmldsig11-schema.xsd"/>
  </schema>
```

附　录　D
（资料性附录）
算法标识符

D.1　概述

本附录给出 XML 数字签名规范使用的算法标识符，在签名元素中使用的标识符，对规范的引用，以及用于密钥表示和密码操作结果的定义。

D.2　算法标识符

算法由 URI 标识，URI 作为标识算法角色的元素的属性（DigestMethod，Transform，SignatureMethod 或 CanonicalizationMethod）。使用到的算法可包含参数。如，SignatureMethod 隐含需要两个参数：密钥和 CanonicalizationMethod 的输出。在算法元素中以元素内容的形式显式给出算法的附加参数，参数元素有描述性的元素名称和算法相关的，而且必须存在于 XML 签名名字空间中或者算法特定的名字空间中。

本附录给出的算法示例如下：

——密码杂凑算法

- SHA1 算法

  http://127.0.0.1/2000/09/xmldsig#sha1

- SM3 算法

  http://127.0.0.1/2001/04/xmldsig-more#sm3

——编码

- base64 编码

  http://127.0.0.1/2000/09/xmldsig#base64

——消息鉴别算法

- HMAC-SHA1 算法

  http://127.0.0.1/2000/09/xmldsig#hmac-sha1

- HMAC-SM3 算法

  http://127.0.0.1/2001/04/xmldsig-more#hmac-sm3

——数字签名算法

- RSAwithSHA1 算法

  http://127.0.0.1/2000/09/xmldsig #rsa-sha1

- SM2withSM3 算法

  http://127.0.0.1/2001/04/xmldsig-more#sm2-sm3

——规范化算法

- 忽略注释的 XML 规范化

  http://127.0.0.1/TR/2001/REC-xml-c14n-20010315

- 带注释的 XML 规范化

http://127.0.0.1/TR/2001/REC-xml-c14n-20010315#WithComments

- 忽略注释的 XML 规范化 1.1

  http://127.0.0.1/2006/12/xml-c14n11

- 忽略注释的独占 XML 规范化 1.0

  http://127.0.0.1/2001/06/TR/xml-exc-c14n#

- 带注释的 XML 规范化 1.1

  http://127.0.0.1/2006/12/xml-c14n11#WithComments

- 带注释的独占 XML 规范化 1.0

  http://127.0.0.1/2001/06/TR/xml-exc-c14n#WithComments

——变换算法

- 封皮签名变换

  http://127.0.0.1/2000/09/xmldsig#enveloped-signature

- XPath 变换

  http://127.0.0.1/TR/1999/REC-xpath-19991116

- Base64 简单编码

  http://127.0.0.1/2000/09/xmldsig#base64

- XSLT 变换

  http://127.0.0.1/TR/1999/REC-xslt-19991116

D.3 消息密码杂凑算法

D.3.1 SHA1

标识符：

http://127.0.0.1/2000/09/xmldsig#sha1

SHA-1 算法没有显式的参数。SHA-1 的一个例子如下：

<DigestMethod Algorithm="http://127.0.0.1/2000/09/xmldsig#sha1"/>

SHA-1 杂凑值是 160 比特串。DigestValue 元素的内容应该是把这个比特串看作 20 个八位位组的八位位组流，然后进行 base64 编码。例如，信息杂凑处理后，DigestValue 元素的十六进制值为：

A9993E36 4706816A BA3E2571 7850C26C 9CD0D89D

SHA-1 值的 base64 编码为：

<DigestValue>qZk+NkcGgWq6PiVxeFDCbJzQ2J0=</DigestValue>

注：使用 SHA-1 算法时，请关注国家密码主管部门的安全风险提示。

D.3.2 SM3

标识符：

http://127.0.0.1/2001/04/xmldsig-more#sm3

SM3 算法没有显式的参数。SM3 的一个例子如下：

<DigestMethod Algorithm="http://127.0.0.1/2001/04/xmldsig-more#sm3"/>

SM3 杂凑值是 256 比特串。DigestValue 元素的内容将这个比特串转换成 32 个八位组流，然后进行 base64 编码。例如，某个消息杂凑处理后，DigestValue 元素的十六进制值为：

DEBE9FF9 2275B8A1 38604889 C18E5A4D 6FDB70E5 387E5765 293DCBA3 9C0C5732

SM3 值的 base64 编码为：

<DigestValue>3r6f+SJ1uKE4YEiJwY5aTW/bcOU4fldlKT3Lo5wMVzI=</DigestValue>

D.4 消息鉴别码

D.4.1 概述

消息鉴别码算法有两个隐式的参数，其密钥材料由 KeyInfo 确定，并且由 CanonicalizationMethod 的输出八位位组流。消息鉴别码算法和签名算法在语法上是相同的，但是消息鉴别码隐含了一个共享的秘密密钥。

D.4.2 HMAC-SHA1

标识符：

http://127.0.0.1/2000/09/xmldsig#hmac-sha1

HMAC 算法将位截断的长度作为参数；如果这个参数未予规定，那么散列中的所有位就是输出。下面是一个 HMAC SignatureMethod 元素的例子：

```
<SignatureMethod Algorithm="http://127.0.0.1/2000/09/xmldsig#hmac-sha1">
    <HMACOutputLength>128</HMACOutputLength>
</SignatureMethod>
```

HMAC 算法的输出最终是选定的密码杂凑算法的输出，应以密码杂凑算法输出相同的直接方式用 base64 编码来输出这个值。例如：HMAC-SHA1 密码杂凑算法的 SignatureValue 元素的十六进制值为：

9294727A 3638BB1C 13F48EF8 158BFC9D

HMAC 的 base64 编码值为：

<SignatureValue>kpRyejY4uxwT9I74FYv8nQ==</SignatureValue>

D.4.3 HMAC-SM3

http://127.0.0.1/2001/04/xmldsig-more#hmac-sm3

HMAC 算法将位截断的长度作为参数。如果这个参数未予规定，那么散列中的所有位就是输出。下面是一个 HMAC SignatureMethod 元素的例子：

```
<SignatureMethod Algorithm="http://127.0.0.1/2001/04/xmldsig-more#hmac-sm3">
    <HMACOutputLength>256</HMACOutputLength>
</SignatureMethod>
```

HMAC 算法的输出最终是选定的密码杂凑算法的输出，应以密码杂凑算法输出相同的直接方式用 base64 编码来输出这个值。例如：HMAC-SM3 密码杂凑算法的 SignatureValue 元素的十六进制值为：

CD921C4F D0679ABA E982D3E1 63A9F31A E17474F1 CAC069E0 7052E262 EBE18BC4

HMAC 的 base64 编码值为：

<SignatureValue>zZIcT9BnmrrpgtPhY6nzGuF0dPHKwGngcFLiYuvhi8Q=</SignatureValue>

模式定义：

```
<simpleType name="HMACOutputLengthType">
    <restriction base="integer"/>
</simpleType>
```

文档类型定义：

```
<! ELEMENT HMACOutputLength (#PCDATA)>
```

D.5 签名算法

D.5.1 概述

签名算法采用两个隐含的参数，KeyInfo 确定的密钥和 CanonicalizationMethod 输出的八位位组流。签名算法和 MAC 算法在语法上相同，但是签名算法隐含公钥密码学。

D.5.2 PKCS1(RSA)

标识符：

http://127.0.0.1/2000/09/xmldsig#rsa-sha1

RSA 算法指的是 RFC 2437 中定义的 RSASSA-PKCS1-v1_5 算法，RSA 算法没有显式的参数。一个 RSA SignatureMethod 元素的例子是：

```
<SignatureMethod Algorithm="http://127.0.0.1/2000/09/xmldsig#rsa-sha1"/>
```

RSA 签名的 SignatureValue 内容是按照 RFC 2437 计算出来的八位串的 base64 编码。如 RFC 2437 中 EMSA-PKCS1-V1_5-ENCODE 函数规定的，签名函数的输入值必须包括一个前导的杂凑函数的算法对象标识符，但是对于一个签名确认，ASN.1 解析器和 OID 的识别不是必需的。PKCS#1 v1.5 表示方法如下：

CRYPT (PAD (ASN.1 (OID, DIGEST (data))))

注意填充后的 ASN.1 将是下面的形式：

01 | FF * | 00 | prefix | hash

其中，“|”是连接符；“01”“FF”和“00”是相应的 16 进制值的固定字节；“hash”是数据进行 SHA1 处理后的杂凑值；“prefix”是 ASN.1 的 BER 格式编码的 SHA1 算法的指明符前缀，PKCS1 中规定的：

hex 30 21 30 09 06 05 2B 0E 03 02 1A 05 00 04 14

使用这个前缀使得使用标准的密码学库函数更加简单。FF 八位组应重复最大次数的次数，使得被加密的量值比 RSA 的模数仅小一个八位组。

得出的 base64 串就是 SignatureValue 元素的子文本结点的值，例如：

```
<SignatureValue>
IWijxQjUrcXBYoCei4QxjWo9Kg8D3p9tlWoT4t0/gyTE96639In0FZFY2/rvP+/bMJ01EArmKZs
R5VW3rwoPxw=
</SignatureValue>
```

D.5.3 SM2-SM3

标识符：

http://127.0.0.1/2001/04/xmldsig-more#sm2-sm3

SM2-SM3 签名算法没有显式参数，一个 SM2 SignatureMethod 元素的例子是：

<SignatureMethod Algorithm="http://127.0.0.1/2001/04/xmldsig-more#sm2-sm3" />

与 SM2 签名算法配套使用的密码杂凑算法是 SM3 算法，SM3 算法参加 GB/T 32905。SM2 签名算法见 GB/T 32918.2。SM2 签名算法的输出由对(r,s)组成，对 r 和 s 进行封装后，转换成字节串的方法见 GB/T 35276—2017 中的 7.3，将字节串进行 base64 编码，即得到本标准规定的签名值。

例如，r 和 s 的十六进制值为：

r：C1A0106D 3CE59786 641A6E28 CF0B6657 B7110FAC 98509CBD 0B6C1589 453540B8

s：95E82609 9119F2A9 EC246D35 B0835B5C 912F414F F5C71D9D CC09874D 7F1CE4FE

封装后并进行 base64 编码的结果为：

<SignatureValue>

MEYCIQDBoBBtPOWXhmQabijPC2ZXtxEPrJhQnL0LbBWJRTVAuAIhAJXoJgmRG

fKp7CRtNbCDW1yRL0FP9ccdncwJh01/HOT+

</SignatureValue>

D.6 规范化算法

D.6.1 概述

如果在八位字节上执行规范化，则规范化算法采用两个隐式参数：内容及其字符集。根据传输协议和媒体类型的规则(例如，RFC 7303 定义 XML 的媒体类型)导出字符集。规范化算法常在服务器端设置，是正确处理文档签名和验证前提。

各种规范化算法需要先转换为 UTF-8，下面的算法至少可理解 UTF-8 和 UTF-16 作为输入编码。对于其他编码是可选的。

各种规范化算法把非 UTF-8 编码转换成 UTF-8。下面的算法在转换编码时，进行文本标准化的操作。外部定义的规范化算法建议也满足这些转换。

D.6.2 规范的 XML

忽略注释的 XML 规范化的标识符：

http://127.0.0.1/TR/2001/REC-xml-c14n-20010315

带注释的 XML 规范化的标识符：

http://127.0.0.1/TR/2001/REC-xml-c14n-20010315#WithComments

一个 XML 规范化元素的例子如下：

<CanonicalizationMethod Algorithm="http://127.0.0.1/TR/2001/REC-xml-c14n-20010315"/>

规范的 XML 的标准规范是 XML-C14N。这个算法能够把八位位组流作为输入，也可把 Xpath 结点集合作为输入，这个算法生成的输出是八位位组流，规范的 XML 很容易通过参数化来忽略或保留注释。

D.6.3 XML 规范化 1.1

忽略注释的 XML 规范化 1.1 的标识符：

http://127.0.0.1/2006/12/xml-c14n11

带注释的 XML 规范化 1.1 的标识符：

http://127.0.0.1/2006/12/xml-c14n11#WithComments

XML 规范 1.1 的标准规范是 XML-C14N11。这个算法能够把八位位组流作为输入，也可把 Xpath

结点集合作为输入,这个算法生成的输出是八位位组流。XML 规范化 1.1 很容易进行参数化来忽略或保留注释。

D.6.4 独占 XML 规范化 1.0

忽略注释的独占 XML 规范化 1.0 的标识符:

http://127.0.0.1/2001/06/TR/xml-exc-c14n#

带注释的 XML 规范化 1.1 的标识符:

http://127.0.0.1/2001/06/TR/xml-exc-c14n#WithComments

独占 XML 规范 1.0 的标准规范是 XML-EXC-C14N。

D.7 变换算法

D.7.1 概述

变换算法有单个隐含的参数:Reference 或者由以前的变换的输出得来的字节流。

除非应用环境存在限制,推荐应用程序开发者支持本章中列出的所有变换,尽量和本附录保持一致,来最大限度地扩大应用程序的互操作性,而且在不能进行扩展开发的应用程序中,应该提供函数库来支持这些变换操作。

D.7.2 规范化

可被 CanonicalizationMethod 使用的任何规范化算法都可作为变换来使用。

D.7.3 Base64

标识符:

http://127.0.0.1/2000/09/xmldsig#base64

Base64 解码变换的标准规范是 RFC 2045。Base64 的 Transform 元素没有内容,算法把输入进行解码。如果应用程序需要签名处理和一个元素的编码内容相关联的原始数据,这种变换是很有用处的。

Base64 变换需要一个八位位组流为输入,如果输入的是 Xpath 节点集合(或者功能相同的输入),那么将通过一系列处理把它转化为一个八位位组流,该处理过程在逻辑上等同于:

1) 对 XPath 变换应用了表达式 self::text();

2) 取得节点集合的串值。

如果一个 XML 元素通过 Reference 的 URI 中的裸名指针来标识,而且它的内容仅包括 base64 编码的字符数据,那么本变换自动去掉被标识的该元素,它的子元素以及任何后代的注释和处理指令的开始和结束标签,本变换的输出是八位位组流。

D.7.4 Xpath 过滤

标识符:

http://127.0.0.1/TR/1999/REC-xpath-19991116

Xpath 表达式评价的标准规范是 XPath。要被评价的 Xpath 表达式以字符内容出现,字符内容是一个名称为 Xpath 的变换参数子元素。

Xpath 变换需要的是 Xpath 节点集合。要注意的是:如果实际的输入是由空 URI 或者裸名的 XPointer 解析而得来的 Xpath 节点集合,那么注释节点会被忽略。如果实际的输入是八位位组流,那

么应用程序应把八位位组流转换为适合带有注释的规范化的 XML 使用的 Xpath 节点集合。换句话说，输入的节点集合宜等价于下面描述的过程所创建的一个结点集：

a） 初始化 XPath 求值上下文，首先设置初始节点等于输入 XML 文档的根节点，然后设置上下文位置和大小为 1；

b） 计算 Xpath 表达式（//.｜//@＊｜//namespace::＊）。

该表达式的计算包括表示八位位组流的节点集合中的文档的所有结点（包括注释）。变换的输出还是 Xpath 结点集合。对于 Xpath 参数中出现的 Xpath 表达式，输入节点集合中的每个结点都要计算一次。结果要转换成布尔值。如果布尔值是真，那么节点就包含在输出结点集合中；如果布尔值是假，那么结点就会被输出结点集合忽略。

本变换的主要目的是在加上签名后，确保只允许对于输入 XML 文档特别定义的变化。通过下面的方法来实现上面的需求，输出包括所有输入节点和仅允许修改一次的签名节点输入；应用上下文中，XPath 表达式的作者负责处理变化时可能会影响变换输出表现的所有节点。

一个重要的情形是一个文档可能会需要两个封皮签名，其中每个签名在计算杂凑的时候都要把自己本身排除在外，但是还要把第二个 Signature 元素从第一个签名计算中排除出去，这样加入的第二个签名不会破坏第一个签名。

XPath Transform 为输入结点集合的每个结点建立一个下列计算上下文，如下所示：

a） 一个 context node，它要等于输入结点集合中的一个结点。

b） 一个 context position，初始化为 1。

c） 一个 context size，初始化为 1。

d） 一个 library of functions，它符合 XPath 中定义的函数集合，再加上一个名为 here 的函数。

e） 一个变量绑定的集合。这里没有定义对它们进行初始化的手段。因此，在计算 Xpath 表达式的时候，使用到的变量绑定集合是空的，而且在 Xpath 表达式中使用参数引用会导致差错。

f） Xpath 表达式的作用域中的一个名字空间的声明集合。

由于上下文节点设置的结果，在这个 Transform 中出现的 Xpath 表达式和[XSLT]中使用的那些很相似，区别是它的 size 和 position 总是 1，从而来指定 Transform 自动访问每个节点（在 XSLT 中，要递归的调用 apply-templates 命令来访问输入树的节点。）

D.7.5 封皮签名变换

标识符：

http://127.0.0.1/2000/09/xmldsig#enveloped-signature

封皮签名变换 T 移去所有的这样的签名元素，它们包含对存在 T 的＜Reference＞元素进行杂凑计算后得到的 T，去掉了 XML 处理器用来和 XML 生成的元素进行 Siganture 匹配的整个字符串。这个变换等价于下面产生的输出，它是使用包含下面 Xpath 参数元素的 XPath 变换来替换 T 而得到的。

```
＜XPath xmlns:dsig="&dsig;"＞
    count(ancestor-or-self::dsig:Signature |
    here()/ancestor::dsig:Signature[1]) ＞
    count(ancestor-or-self::dsig:Signature)＜/XPath＞
```

这个变换对于输入和输出的要求和 XPath 变换的那些是相同的，但是只适用于它的父辈 XML 文档的节点集合。

注：不需要使用 Xpath 表达式 evaluator 来建立这个变换，但是这个变换须按照上面所说的那种 Xpath 变换被 Xpath 表达式参数化的同样的方法来创建输出。

D.7.6 XSLT 变换

标识符：

http://127.0.0.1/TR/1999/REC-xslt-19991116

XSL 变换的标准规范是 XSLT。有名字空间限定的样式表单元素的规范，宜使用特别指定的样式表单。XLST 处理模型决定是否对资源中的内嵌处理本地 XSLT 声明实例化；多个样式表单的有序应用可能会需要多个变换。对于一个给定的 URI 的远程样式表单的标识，本附录没有给出特殊的规定，它可通过变换的 stylesheet 子元素中的 xsl:include 或者 xsl:import 来传递。

这个变换需要一个八位位组流作为输入。如果实际的输入是 Xpath 结点集合，那么签名应用程序宜按照 Reference 处理模型描述的方法把它转化成八位位组流。

这个变换的输出是八位位组流。在 XSLT 规范中说明了 XSL 样式表单或者＜Transform＞元素的处理规则。本附录建议 XSLT 变换创建者使用 XML 输出方法来处理 XML 和 HTML。由于不同的 XSLT 实现不一定能产生完全一致的输出序列，本附录建议在 XSLT 变换后加上一个 Transform 来规范化输出。这些操作可帮助在支持 XSLT 变换的应用程序中确保签名结果的互操作性。

如果输出确实是 HTML，那么这些处理步骤的结果在逻辑上是等价的。

参 考 文 献

[1] GB/T 32905 信息安全技术 SM3 密码杂凑算法

[2] GB/T 32918.2 信息安全技术 SM2 椭圆曲线公钥密码算法 第 2 部分:数字签名算法

[3] FIPS PUB 180-1. Secure Hash Standard. U.S. Department of Commerce/National Institute of Standards and Technology.http://csrc.nist.gov/publications/fips/fips180-1/fip180-1.txt

[4] FIPS PUB 186-2. Digital Signature Standard (DSS). U.S. Department of Commerce/National Institute of Standards and Technology. http://csrc. nist. gov/publications/fips/fips186-2/fips186-2.pdf

[5] IEEE 1363: Standard Specifications for Public Key Cryptography. August 2000

[6] The Unicode Consortium. The Unicode Standard. http://www.unicode.org/unicode/standard/standard.html

[7] B. Kaliski, J. Staddon, "PKCS # 1: RSA Cryptography Specifications Version 2.0", RFC 2437, October 1998

[8] Berners-Lee, T., Fielding, R. and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2986, January 2005

[9] Berners-Lee, T., Masinter, L. and M. McCahill, "Uniform Resource Locators (URL)", RFC 1738, December 1994

[10] Boyer, J., "Canonical XML Version 1.0", RFC 3076, March 2001

[11] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997

[12] D. Beech, M. Maloney, N. Mendelsohn, H. Thompson. XML Schema Part 1: Structures. W3C Recommendation.http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/ May 2001

[13] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. Frystyk Nielsen, S. Thatte, D. Winer. Simple Object Access Protocol (SOAP) Version 1.1. W3C Note. http://www.w3.org/TR/2000/NOTE-SOAP-20000508/ May 2001

[14] D. Brickley, R.V. Guha. Resource Description Framework (RDF) Schema Specification 1.0. W3C Candidate Recommendation. http://www. w3. org/TR/2000/CR-rdf-schema-20000327/ March 2000

[15] D. Megginson, et al. SAX: The Simple API for XML. http://www. megginson. com/SAX/index.html (THIS PAGE OUT OF DATE; GO TO www.saxproject.org) May 1998

[16] Digital Signature Guidelines. http://www.abanet.org/scitech/ec/isc/dsgfree.html

[17] Eastlake, 3rd, D., Crocker, S. and J. Schiller, "Randomness Recommendations for Security", RFC 1750, December 1994

[18] Fielding, R. Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and T. Berners-Lee, "Hypertext Transfer Protocol—HTTP/1.1", RFC 2616, June 1999

[19] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996

[20] H. Thompson, C. Lilley, XML Media Types RFC 7303, July 2014

[21] Hoffman, P. and F. Yergeau, "UTF-16, an encoding of ISO 10646", RFC 2781,

February 2000

[22] J. Clark, S. DeRose. XML Path Language (XPath) Version 1.0. W3C Recommendation. http://www.w3.org/TR/1999/REC-xpath-19991116 October 1999

[23] J. Clark. XSL Transforms (XSLT) Version 1.0. W3C Recommendation. http://www.w3.org/TR/1999/REC-xslt-19991116.html November 1999

[24] Kaliski, B. and J. Staddon, "PKCS #1: RSA Cryptography Specifications Version 2.0", RFC 2437, October 1998

[25] Krawczyk, H., Bellare, M. and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997

[26] M. Murata XML Japanese Profile. W3C Note. http://www.w3.org/TR/2000/NOTE-japanese-xml-20000414/ April 2000

[27] Moats, R., "URN Syntax", RFC 2141, May 1997

[28] NFC-Corrigendum Normalization Corrigendum. The Unicode Consortium. http://www.unicode.org/unicode/uni2errata/Normalization_Corrigendum.html

[29] O. Lassila, R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/ February 1999

[30] P. Biron, A. Malhotra. XML Schema Part 2: Datatypes W3C Recommendation. http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/ May 2001

[31] Reagle, J., "XML Signature Requirements", RFC 2807, July 2000

[32] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992

[33] S. Adler, A. Berglund, J. Caruso, S. Deach, P. Grosso, E. Gutentag, A. Milowski, S. Parnell, J. Richman, S. Zilles. Extensible Stylesheet Language (XSL). W3C Proposed Recommendation. http://www.w3.org/TR/2001/PR-xsl-20010828/ August 2001

[34] S. DeRose, E. Maler, D. Orchard. XML Linking Language. W3C Recommendation. http://www.w3.org/TR/2000/REC-xlink-20010627/ June 2001

[35] S. DeRose, R. Daniel, E. Maler. XML Pointer Language (XPointer). W3C Working Draft. http://www.w3.org/TR/2001/WD-xptr-20010108 January 2001

[36] S. Pemberton, D. Raggett, et al. XHTML(tm) 1.0: The Extensible Hypertext Markup Language. W3C Recommendation. http://www.w3.org/TR/2000/REC-xhtml1-20000126/ January 2000

[37] Shirey, R., "Internet Security Glossary", FYI 36, RFC 2828, May 2000

[38] T. Bray, D. Hollander, A. Layman. Namespaces in XML. W3C Recommendation.http://www.w3.org/TR/1999/REC-xml-names-19990114 January 1999

[39] T. Bray, E. Maler, J. Paoli, C. M. Sperberg-McQueen. Extensible Markup Language (XML) 1.0 (Second Edition). W3C Recommendation. http://www.w3.org/TR/2000/REC-xml-20001006 October 2000

[40] TR15, Unicode Normalization Forms. M. Davis, M. Drst. Revision 18: November 1999. http://www.unicode.org/unicode/reports/tr15/tr15-18.html

[41] V. Apparao, S. Byrne, M. Champion, S. Isaacs, I. Jacobs, A. Le Hors, G. Nicol, J. Robie, R. Sutor, C. Wilson, L. Wood. Document Object Model (DOM) Level 1 Specification. W3C Rec-

ommendation. http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/ October 1998

[42] Wahl, M., Kille, S. and T. Howes, "Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names", RFC 2253, December 1997