

Review of

# XML Signatures

W3C - XML Signature Syntax and Processing  
&  
RFC3275

Markus Weissmann, <[mail@mweissmann.de](mailto:mail@mweissmann.de)>

15.06.2005

# Introduction

## The XML Signature Standard

The "XML Signature Syntax and Processing"<sup>1</sup> standard is a recommendation of the W3C<sup>2</sup> and also a RFC<sup>3</sup> of the IETF<sup>4</sup>. It proposes an open standard for how to cryptographically sign XML<sup>5</sup> documents, URIs<sup>6</sup> and derived objects. XML signature is a digital signature method that takes place on the presentation or 6th layer of the OSI model. The revision from 12. february 2002 is the basis for the following considerations.

## Problems to be solved

With current protocols there are specifically three problems that XML signature can solve. The persistence of digital signatures, the possibility for multiple persons to sign the same XML document and of course to create an open, platform independent standard how to sign XML data.

- The common online B2C shop, like amazon or ebay, do not let their customers digitally sign the contracts on which they agree on. A SSL encrypted HTTP session and a username/password authentication are used to place orders. As soon as the session ends - normally shortly after the user has placed his order - the cryptographic protection of the contract comes to its end. This leaves the contract vulnerable to changes, which becomes a problem when the database server gets compromised. An intruder or even the company running the server do have access to the orders and may adjust them to their advantage.
- When signing a document with e. g. PGP<sup>7</sup>, the resulting document includes the signature. If this exact document is to be signed again - by a second party - you can either make a copy of the original document and sign that, or sign the derived document. The second possibility leaves you with the problem of comparing the signed documents, the first one is problematic if the party that signed the document first withdraws its signature (due to key expiration, bankrupt of the company, etc.). If this happens, it is unclear what the duties of the second party are; after all the document they signed has meanwhile become invalid.
- The most commonly used methods for signing digital documents are probably PGP and S/MIME. Both signature functions rely on some kind of MIME encapsulation. This makes using them kind of clumsy in conjunction with XML documents. To sign a XML document, you first have to create a text representation from the XML tree, sign this text document

---

<sup>1</sup> XML Signature Syntax and Processing, <http://www.w3.org/TR/xmlsig-core/>

<sup>2</sup> World Wide Web Consortium, <http://www.w3.org/>

<sup>3</sup> RFC 3275, XML-Signature Syntax and Processing, <http://www.ietf.org/rfc/rfc3275.txt>

<sup>4</sup> Internet Engineering Task Force, <http://www.ietf.org/>

<sup>5</sup> Extensible Markup Language, <http://www.w3.org/XML/>

<sup>6</sup> RFC 2396, Uniform Resource Identifiers, <http://www.ietf.org/rfc/rfc2396.txt>

<sup>7</sup> RFC 2440, OpenPGP Message Format, <http://www.ietf.org/rfc/rfc2440.txt>

and then create some kind of BLOB from it which is to be inserted in an encapsulating XML document. When checking the signature or when trying to get the contents of the signed XML document, you always have to extract and parse the wrapped document. This behavior is slow, error-prone, tends to produce unnecessarily large documents and makes using query languages - which operate directly on XML trees - like XPath<sup>8</sup> or XQuery<sup>9</sup> futile.

# XML Signatures

## Location in OSI stack

The signing of XML documents takes place in the presentation layer of the OSI reference layer, the application layer of the TCP/IP protocol respectively. This has several consequences:

- The signature does not get dropped when the transport layer is left, like it is with SSL/TLS after a TCP session ends.
- There does not have to be a network connection at all; you may sign XML documents and just save them to your hard-disk etc.. XML Signatures can be used off-line, too.

Strictly speaking, XML signature is also independent of the application, which makes it possible to exchange signed documents between XML signature enabled applications. You could sign the XHTML<sup>10</sup> content of an email message with your mail program, while the recipient may have a server that is running a web-mailer check the signature for validity. Nevertheless, it is not enough for an application to just check every XML data it receives for signatures. There also has to be a set of rules what to do with expired as it has to inform its user about the validity of the signature and chose actions accordingly.

## Signature Composition

A XML signature is itself a XML document. It is a compound of four objects, two of which are mandatory.

1. The “SignedInfo” sub-document specifies the data to be signed.
2. “SignatureValue” yields the digital signature itself.
3. An optional “KeyInfo” node contains information about the key used to sign the document.
4. A list of “Object” nodes completes the document. Note that this list may just have any number of elements, though may as well be empty.

## SignedInfo

The “SignedInfo” sub-tree contains itself three objects:

---

<sup>8</sup> XML Path Language (XPath), <http://www.w3.org/TR/xpath>

<sup>9</sup> XML Query Language (XQuery), <http://www.w3.org/TR/xquery>

<sup>10</sup> Extensible HyperText Markup Language, 2nd Edition, <http://w3.org/TR/xhtml1>

1. The “CanonicalizationMethod”, giving the method used for the XML canonicalization. This function is important as equivalency of XML documents can not be obtained easily. The XML standard<sup>11</sup> leaves some room to specify the same content in multiple ways. For example the usage of quotes is not fixed as whether to use single or double quotes around an attribute value. Also the order of attributes within the same XML node does must not be taken into account when obtaining information from a XML document. The “SignedInfo” object must therefore specify the algorithm that was used to bring the signed document into the canonical version. Currently the standard only allows a single canonicalization algorithm<sup>12</sup> of the W3C.
2. The “SignatureMethod” gives the algorithms used to obtain the signature. This may be DSA with SHA1, RSA with SHA1 or HMAC<sup>13</sup> with SHA1. As can easily be obtained from this list, the only available hash algorithm is SHA1<sup>14</sup>, while multiple cryptographic signature algorithms are allowed. It is of course be foreseeable that this list will be extended with newer algorithms.  
The XML signature standard itself does not declare the mathematical signature functions like RSA, but relies on them being common knowledge.
3. The 3rd part of the “SignedInfo” element is a list of “Reference” objects. This list has to have at least one element and may not be empty. A “Reference” does exactly that, reference a document to be signed. There is a plethora of ways to declare the document to sign, the simplest of is just a simple URI. Specifying a URI in the “Reference” makes the signature a so called “detached signature”, it signs content that is not part of the same document as itself. When using a XML id to specify a sub-tree inside the same document, you get an enveloped signature.  
Also in the “Reference” section is the “Digest”, declaring the hash algorithm used to create the digest that finally is signed. As the MD5 algorithm is clearly at the end of its life, the W3C chose to use SHA1 just. More recent algorithms can be expected to be allowed as they get adopted by the industry.  
An optional part of the “Reference” is a “Transform” object. Here you may give further advice what part of the referenced XML document you wish to sign. The rules have to be given as either XPath or XSL<sup>15</sup> code. While XPath is just a query language for XML, XSL is a full blown functional programming language. This can make it hard to impossible to check a signature, as XSL is turing complete<sup>16</sup> which brings along the halting problem. Therefore it is advisable to implement the signature verification as a sandboxed system that will get a concrete amount of CPU time only.

## SignatureValue

The “SignatureValue” XML node is a very simple object: It contains just the base-64 encoded value of the digital signature.

---

<sup>11</sup> Extensible Markup Language, <http://www.w3.org/XML>

<sup>12</sup> Canonical XML 1.0, <http://www.w3.org/TR/2001/REC-xml-c14n-20010315>

<sup>13</sup> RFC 2104, HMAC: Keyed-Hashing for Message Authentication, <http://www.ietf.org/rfc/rfc2104.txt>

<sup>14</sup> Secure Hash Standard, <http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf>

<sup>15</sup> Extensible Stylesheet Language, <http://www.w3.org/Style/XSL/>

<sup>16</sup> Universal Turing Machine in XSLT, <http://www.unidex.com/turing/utm.htm>

## KeyInfo

This optional node holds information about the key used to create the signature. Information about the owner of the digital key etc. can be stored here.

## Objects

The optional list of “Object” nodes contains additional references to be signed. Popular example for this feature is to embed an additional time-stamp to a signature to exactly specify the time when the signature has been created. There is an abundance of usages for this, for example to give the location where the document has been signed, the time-zone if the given time has not been agreed on to be e. g. UTC. and so forth.

## Signature Computation, Verification

To check a XML signature, it is necessary to evaluate the “SignedInfo” object and compute it’s value. The resulting document is then being put into a cryptographic hash function. The result of this operation - in case of SHA1 a 160 byte array - is then used as input for the signature function. If the result of this operation and the value of the “SignatureValue” element fit together for the given signature key, the signature is valid.

Creating the signature follows the same steps, but using the private key computes the value of the “SignatureValue” element from the 160 bytes digest instead.

## Current Implementations

Currently there is already an impressive list of available implementations of the XML signature standard:

- The Apache Foundation’s<sup>17</sup> XML Security group<sup>18</sup> works on both, a Java and a C++ implementation of XML signature and XML encryption. The Java framework builds on Apache’s Xerces<sup>19</sup> and Xalan<sup>20</sup> libraries and needs - due to U.S. export restrictions - an additional JCE library like the one from Bouncy Castle<sup>21</sup>. The C++ version also makes use of Xalan and Xerces for XML processing.
- From Sun Microsystems there is the “Web Services Developer Pack” that makes use of the Apache Foundation’s Java code-base. and gives it Sun’s blessing as it is made available in the Sun reserved namespace javax.xml.crypto.dsig.
- For the C programming language there is an open source implementation by the XMLSec project<sup>22</sup>. This is a community based effort that depends on the widespread

---

<sup>17</sup> Apache Foundation, <http://www.apache.org/>

<sup>18</sup> Apache XML Security, <http://xml.apache.org/security/>

<sup>19</sup> Apache Xerces-J Java XML parser, <http://xml.apache.org/xerces-j/>

<sup>20</sup> Apache Xalan-J XSLT processor, <http://xml.apache.org/xalan-j/>

<sup>21</sup> Legion of Bouncy Castle, <http://www.bouncycastle.org/>

<sup>22</sup> XML Security Library, <http://www.aleksey.com/xmlsec/>

libxml and libxslt libraries<sup>23</sup> developed by the GNOME foundation<sup>24</sup>. This library is available on most Unix-like operating systems, e. g. runs on FreeBSD, Linux, Mac OS X and Solaris.

- Building on XMLSec a Python language<sup>25</sup> extension is being under development. The PyXMLSec<sup>26</sup> GPL licensed implementation currently already makes about 300 of XMLSec's function calls available in the Python context. As the Python interpreter runs on almost all major platforms, from cell phones<sup>27</sup> to multiprocessor Unix servers, the availability of PyXMLSec is just bound to the platforms that XMLSec supports.
- The Microsoft .NET class library also features functions for XML signatures. It is available in the System.Security.Cryptography.Xml namespace. It is questionable if it is wise to use a closed source framework for cryptographic functions - especially if there are compatible open source implementations at hand. Another drawback of this framework is, that Microsoft's .NET virtual machine will run on Microsoft Windows only.
- The Mono project<sup>28</sup> that tries to create an open source implementation of Microsoft's .NET virtual machine and class library, also yields the same classes and functions for XML signatures as the archetype does. The Mono VM is compatible with most operating systems (Linux, BSD, Solaris, Windows, Mac OS X) and architectures (PowerPC, IA32, AA64, SPARC32, SPARC64).

The current implementations of the XML signature standard appear to be pretty promising, with probably most of them being in a late beta phase. On practically all major platforms you may choose between at least three of them.

It remains to be seen if the XML signature standard is strict enough already to guarantee interoperability between the implementations. One can expect various smaller problems that need to be fixed.

## Shortcomings of XML Signatures

Currently there are still some problems with XML signature some of which may even deny the big break-through to the standard.

The most obvious and most challenging problem seems to be the lack of public/private key infrastructure with the end user. This problem is also holding back the broad usage of email-encryption and other encrypted communication. It seems as if XML signature will not change anything here, as is obviously not a technical problem but rather a question of acceptance and legislative decisions. Email-signing will not see broader acceptance, just because it now uses a XML based signature function; it is way too complicated for the average user to obtain a public/private key pair to do his online shopping, especially if it used

---

<sup>23</sup> The XML C parser and toolkit of Gnome, <http://www.xmlsoft.org/>

<sup>24</sup> GNOME Foundation, <http://www.gnome.org/>

<sup>25</sup> Python Programming Language, <http://www.python.org/>

<sup>26</sup> PyXMLSec - XMLSec Python bindings, <http://pyxmlsec.labs.libre-entreprise.org/>

<sup>27</sup> Python for Nokia Series 60, <http://www.forum.nokia.com/main/0,,034-821,00.html>

<sup>28</sup> Mono Project, <http://www.mono-project.com/>

to work without before. The two most likely candidates to change this public disinterest in cryptography are online banking and e-government.

From a pure technical perspective, the biggest drawback of XML signature is the use of XSL. As XSL sports  $\mu$ -recursion, it is turing complete, bringing along the halting problem. This makes it obviously impossible to determine if a given XSL program will ever halt. This fact can easily bring down a publicly available B2C or B2B server in a denial of service attack, but also stack overflows, brought along by unrestricted recursions, could be possible. To obtain a controlled behavior, one can either abstain from XSL altogether or at least perform the evaluation inside a sandbox.

## Future of XML Signatures

The W3C recommendation for signing XML data appears to be well thought through; the plenitude of implementations back up this impression. This will make it hard to impossible for a competing XML signature standard to succeed, given that not only IT heavy weights like Microsoft, Sun and Novell<sup>29</sup> (which supports the mono project) but also reputable groups like the Apache and GNOME foundations support RFC 3275.

XML Signature may well succeed both PGP and S/MIME as the standard for cryptographically signing emails, but more likely will be successful in newly developed software or systems that already make use of XML - which are quite a lot already.

## Conclusions

XML Signatures are a hot topic currently, though this may well be due to the still ongoing XML hype. Nevertheless, XML has meanwhile become indispensable in practically all modern software systems. Digital signatures also have become more common, from email to signed executables or source code, the most prominent one probably being the signed drivers that Microsoft tries to make the default for their latest desktop operating systems (set aside the SSL certificates that web-browsers and servers use and most users tend to ignore anyway).

The common need for digital security, that grew during the last 10 years due to the emergence of the internet and the e-commerce platforms this brought along, paired with the de facto standard for computer data organization obviously had to constitute some kind of XML signature. RFC 3275 can not deny its origin as recommendation from the creators of XML and keepers of the world wide web.

The XML signature standard will see widespread use in server applications, where security beyond sessions is important, like is in B2B systems. The common end-user may well never get into contact with XML signatures due to acceptance problems most cryptographic applications do have.

---

<sup>29</sup> Novell, <http://www.novell.com/>