Enabling XML security

An introduction to XML encryption and XML signature

Murdoch Mactaggart (IBMDev@TextBiz.com)
Freelance author
September 2001

XML is a major enabler of what the Internet, and latterly Web services, require in order to continue growing and developing. Yet a lot of work remains to be done on security-related issues before the full capabilities of XML languages can be realised. At present, encrypting a complete XML document, testing its integrity, and confirming the authenticity of its sender is a straightforward process. But it is increasingly necessary to use these functions on parts of documents, to encrypt and authenticate in arbitrary sequences, and to involve different users or originators. At present, the most important sets of developing specifications in the area of XML-related security are XML encryption, XML signature, XACL, SAML, and XKMS. This article introduces the first two.

Introduction

XML has become a valuable mechanism for data exchange across the Internet. SOAP, a means of sending XML messages, facilitates process intercommunication in ways not possible before, while UDDI seems to be fast becoming the standard for bringing together providers and users of Web services; the services themselves are described by XML in the form of WSDL, the Web Services Description Language. Without XML, this flexibility and power would not be possible and, as various people have remarked, it would be necessary to invent the metalanguage.

The other area of rapid growth is that of security. Traditional methods of establishing trust between parties aren't appropriate on the public Internet or, indeed, on large LANs or WANs. Trust mechanisms based on asymmetric cryptography can be very useful in such situations, but the ease of deployment and key management, the extent of interoperability, and the security offered are, in reality, far less than the enthusiastic vendors of different Public Key Infrastructures (PKI) would have us believe. There are particular difficulties in dealing with hierarchical data structures and with subsets of data with varying requirements as to confidentiality, access authority, or integrity. In addition, the application of now standard security controls differentially to XML documents is not at all straightforward.

Several bodies are actively involved in examining the issues and in developing standards. The main relevant developments here are XML encryption and the related XML signature, eXtensible Access Control Language (XACL), and the related Security Assertion Markup Language (SAML -- a blending of the formerly competing AuthML and S2ML). Each of these is driven by OASIS, and XML Key Management Specification (XKMS). This article introduces XML encryption and XML signature.

XML encryption and XML signature

An XML document, like any other, can be encrypted in its entirety and sent securely to one or more recipients. This is a common function of SSL or TLS, for example, but what is much more interesting is how to handle situations where different parts of the same document need different treatment. A valuable benefit of XML is that a complete document can be sent as one operation and then held locally, thus reducing network traffic. But this then raises the question of how to control authorised viewing of different groups of elements. A merchant may need to know a customer's name and address but doesn't need to know the various details of any credit card being used any more than the bank needs to know the details of the goods bought. A researcher may need to be prevented from seeing personal details on medical records while an administrator may need exactly those details but should be prevented from viewing medical history; a doctor or nurse, in turn, may need medical details and some, but not all, personal material.

Cryptography now does far more than merely conceal information. Message digests confirm text integrity, digital signatures support sender authentication, and related mechanisms are used to ensure that a valid transaction cannot later be repudiated by another party. These are all essential elements of remote trading, and mechanisms for handling complete documents are now fairly well-developed.

As with general encryption, there's no problem in digitally signing an XML document as a whole. However, difficulty arises when parts of a document need to be signed, perhaps by different people, and when this needs to be done in conjunction with selective encryption. It may not be possible or desirable to mandate a particular sequence of sectional encryption by specified people acting in order, yet successful processing of the different parts of the document will depend on knowing this. Further, as a digital signature asserts that a certain private key has been used to authenticate something, it's prudent that a signer view the item to be signed in plain text, and this may mean decrypting part of something already encrypted for other reasons. In other cases, data that is already encrypted may be encrypted further as part of a larger set. The more the different possibilities are considered in sets of transactions involving a single XML document -- perhaps a Web form or a series of data records used in a workflow sequence, processed by a number of different applications and different users -- the more the huge potential complexity can be seen.

There are additional problems, as well. One of the strengths of XML languages is that searching is clear and unambiguous: The DTD or schema provides information as to the relevant syntax. If a document subsection, including tags, is encrypted as a whole, then the ability to search for data relevant to those tags is lost. Further, if the tags are themselves encrypted, then, being known, they may be useful as material for mounting plain text attacks against the cryptography employed.

These are some of the areas that the working groups are considering.

XML Security Suite
In part because the standards are still developing, the number of toolkits and libraries available to developers are still limited, although this is certainly beginning to change. IBM has submitted two relevant Java Specification Requests (JSRs) to the Java Community Process (JCP). These are JSR-105, XML Digital Signature APIs, and JSR-106, Digital Encryption APIs.

The IBM Tokyo Research Laboratory developed the XML Security Suite in 1999 as a prototype implementation of XML signature. It contains utilities that automatically generate XML digital signatures, implement the W3C's Canonical XML working draft, and provide element-level encryption through an experimental implementation of XML encryption. It also provides a means of dealing with the particular requirements of security as they apply to XML documents. The XML schema definition of the eXtensible Access Control Language (XACL) is also introduced.

developerWorks has a detailed article on the suite by Doug Tidwell, and the latest version of the suite itself is available on the alphaWorks site. (See Resources.)

XML encryption examples

The core element in the XML encryption syntax is the `EncryptedData` element which, with the `EncryptedKey` element, is used to transport encryption keys from the originator to a known recipient, and derives from the `EncryptedType` abstract type. Data to be encrypted can be arbitrary data, an XML document, an XML element, or XML element content; the result of encrypting data is an XML encryption element that contains or references the cipher data. When an element or element content is encrypted, the `EncryptedData` element replaces the element or content in the encrypted version of the XML document. When it's arbitrary data that is being encrypted, the `EncryptedData` element may become the root of a new XML document or it may become a child element. When an entire XML document is encrypted, then the `EncryptedData` element may become the root of a new document. Further, `EncryptedData` cannot be the parent or child of another `EncryptedData` element, but the actual data encrypted can be anything including existing `EncryptedData` or `EncryptedKey` elements.

The encryption working draft gives examples of how the granularity of encryption may differ according to different requirements and what the consequences might be. The code fragment in Listing 1 shows an unencrypted XML document with credit card and other personal information. In some cases (for example, concealing information on payment mechanisms) it may be desirable to encrypt everything other than the customer name, and the code fragment in Listing 2 shows how this can be done.

Listing 1. Information on John Smith showing his bank, limit of $5,000, card number, and expiration date

```
        <?xml version='1.0'?>
        <PaymentInfo xmlns='http://example.org/paymentv2'>
          <Name>John Smith<Name/>
          <CreditCard Limit='5,000' Currency='USD'>
            <Number>4019 2445 0277 5567</Number>
            <Issuer>Bank of the Internet</Issuer>
            <Expiration>04/02</Expiration>
          </CreditCard>
        </PaymentInfo>
```

Listing 2. Encrypted document where everything other than the name is encrypted

```
        <?xml version='1.0'?>
        <PaymentInfo xmlns='http://example.org/paymentv2'>
          <Name>John Smith<Name/>
          <EncryptedData Type='http://www.w3.org/2001/04/xmlenc#Element'
           xmlns='http://www.w3.org/2001/04/xmlenc#'>
              <CipherData><CipherValue>A23B45C56</CipherValue></CipherData>
          </EncryptedData>
        </PaymentInfo>
```

In yet other cases, it might only be necessary to conceal some sensitive content -- perhaps from a merchant or other third party -- and Listing 3 shows this. (Note that the tag name relevant to the encrypted content is shown.)

Listing 3. Encrypted document with only the credit card number hidden

```
        <?xml version='1.0'?>
        <PaymentInfo xmlns='http://example.org/paymentv2'>
          <Name>John Smith<Name/>
          <CreditCard Limit='5,000' Currency='USD'>
            <Number>
              <EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
               Type='http://www.w3.org/2001/04/xmlenc#Content'>
                  <CipherData><CipherValue>A23B45C56</CipherValue>
                  </CipherData>
              </EncryptedData>
            </Number>
            <Issuer>Bank of the Internet</Issuer>
            <Expiration>04/02</Expiration>
          </CreditCard>
        </PaymentInfo>
```

It might also be necessary to encrypt *all* information in the document and Listing 4 shows this.

Listing 4. Encrypted document with the complete contents hidden

```
<?xml version='1.0'?>
       <EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
        Type='http://www.isi.edu/in-notes/iana/assignments/media-types/text/xml'>
               <CipherData><CipherValue>A23B45C56</CipherValue></CipherData>
       </EncryptedData>
```

The `CipherData` element can either envelop or reference the raw encrypted data. In the first case, that raw data is shown by the contents of the `CipherValue` element, while in the second a `CipherReference` element is used, and this encloses a

URI which points to the location of the encrypted data.

Canonical XML

The slightest change to the message to which a cryptographic hash algorithm is applied will result in a different value. This provides confidence as to the integrity of the message and is fine for normal use, but introduces a further complication -- two XML documents may differ in exact textual comparison though they may be logically equivalent. Matters such as line delimiters, empty tags, use of hex values instead of names in attributes, and presence or variation of comments in certain situations are all instances where one document may differ from another without the logical structure being affected. The canonical XML specification describes a method for generating a physical representation of a document, known as the canonical form, that accounts for permissible variations such that if two documents have the same canonical form, then the two documents are to be considered logically equivalent within a given application context.

This is particularly important so far as encryption and, more specifically, digital signing are concerned because it's clearly necessary that textual variants that have no logical change implications should not suggest that the integrity of a document or the authentication of its sender is suspect. This is something that might otherwise happen as a result of different treatment by different tools (such as parsers) generating different texts and, in turn, different message digests. Hence, during both signature generation and validation computing, the message digest should be done on the canonical form. If the digests match, then this confirms that the canonical forms over which they were computed also match, even though the textual forms may differ.

XML signature examples

XML signatures can be applied to any arbitrary data content. Those that are applied to data within the same XML document as the signature are termed *enveloping* or *enveloped* signatures while those in which the data is external to the signature element are termed *detached* signatures. Listing 5, taken from the signature candidate recommendation document, is an instance of a simple detached signature.

Listing 5. Example of a simple detached signature

```
[s01] <Signature Id="MyFirstSignature"
      xmlns="http://www.w3.org/2000/09/xmldsig#">
[s02]   <SignedInfo>
[s03]   <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/
        REC-xml-c14n-20010315"/>
[s04]   <SignatureMethod Algorithm="http://www.w3.org/2000/09/
        xmldsig#dsa-sha1"/>
[s05]   <Reference URI="http://www.w3.org/TR/2000/REC-xhtml1-20000126/">
[s06]     <Transforms>
[s07]       <Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
            20010315"/>
[s08]     </Transforms>
[s09]     <DigestMethod Algorithm="http://www.w3.org/2000/09/
             xmldsig#sha1"/>
[s10]     <DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
[s11]   </Reference>
[s12] </SignedInfo>
[s13]   <SignatureValue>MC0CFFrVLtRlk=...</SignatureValue>
[s14]   <KeyInfo>
[s15a]    <KeyValue>
[s15b]      <DSAKeyValue>
[s15c]        <p>...</p><Q>...</Q><G>...</G><Y>...</Y>
[s15d]      </DSAKeyValue>
[s15e]    </KeyValue>
[s16]   </KeyInfo>
[s17] </Signature>
```

The information that is actually signed is that between lines s02 and s12, the `SignedInfo` element. Reference to the algorithms used in calculating the `SignatureValue` element is included within the signed section while that element itself is outside the signed section, on line s13. The `SignatureMethod` reference on line s04 is to the algorithm used to convert the canonicalized `SignedInfo` into the `SignatureValue`. It's a combination of a key-dependent algorithm and a digest algorithm, here DSA and SHA-1, possibly with other manipulation such as padding. The `KeyInfo` element (here lines s14 to s16 -- this element is optional) indicates the key that's used to validate the signature.

Transforms
As mentioned earlier, there is a wide range of possibilities as to the order in which encryption, signing, modifying, and perhaps more signing may take place. Users may be required to enter more data into a field of a form that is already partially encrypted or partially signed, and need to be able to do so without preventing later validation and decryption. The W3C recently published a working draft on decryption transform for XML signature that addresses this situation. (See Resources.)

The example below, taken from that document, shows how the recipient of a document can be advised as to the proper order of decryption and signature verification. The first fragment shows the part of the document to be signed, the `order` element; within this, the personal and financial details of the `cardinfo` element, lines 7 to 11, are in clear text but some encrypted data is already present (line 12).

Listing 6. Order element within an XML document

```
          [01] <order Id="order">
          [02]   <item>
          [03]     <title>XML and Java</title>
          [04]     <price>100.0</price>
          [05]     <quantity>1</quantity>
          [06]   </item>
          [07]   <cardinfo>
          [08]     <name>Your Name</name>
          [09]     <expiration>04/2002</expiration>
          [10]     <number>5283 8304 6232 0010</number>
          [11]   </cardinfo>
          [12]   <EncryptedData Id="enc1"xmlns="http://www.w3.org/
                 2001/04/xmlenc#">...</EncryptedData>
          [13] </order>
```

Listing 7. Order document after signing and further encrypting and now showing transform information

```
          [01] <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
          [02]   <SignedInfo>
          [03]     ...
          [04]     <Reference URI="#order">
          [05]       <Transforms>
          [06]         <Transform Algorithm="http://www.w3.org/2001/04/
xmlenc#decryption">
          [07]           <DataReference URI="#enc1"
                          xmlns="http://www.w3.org/2001/04/xmlenc#"/>
          [08]         </Transform>
          [09]         <Transform Algorithm="http://www.w3.org/TR/2000/
                       CR-xml-c14n-20001026"/>
          [10]       </Transforms>
          [11]       ...
          [12]     </Reference>
          [13]   </SignedInfo>
          [14]   <SignatureValue>...</SignatureValue>
          [15]   <Object>
          [16]     <order Id="order">
          [17]       <item>
          [18]         <title>XML and Java</title>
          [19]         <price>100.0</price>
          [20]         <quantity>1</quantity>
          [21]       </item>
```

```
[22]       <EncryptedData Id="enc2"
           xmlns="http://www.w3.org/2001/04/xmlenc#">...</EncryptedData>
[23]       <EncryptedData Id="enc1"
           xmlns="http://www.w3.org/2001/04/xmlenc#">...</EncryptedData>
[24]     </order>
[25]   </Object>
[26] </Signature>
```

The `Signature` element, lines 1 to 26, now includes the previous `order` element, lines 16 to 24, with its earlier plain text `cardinfo` element encrypted and shown in the single line 22. There are two transform references: decryption (lines 6 to 8) and canonicalization (line 9). The decryption transform instructs the signature verifier to decrypt all the encrypted data except for the one specified on line 7 in the `DataRef` element. After the `EncryptedData` element in line 22 is decrypted, the `order` element is canonicalized and the signature duly verified.

Other relevant languages and specifications

Concealing sensitive information within XML documents, establishing integrity, and authenticating the source of different parts of such documents is largely handled by following the procedures set out in the encryption and signature specifications described in the W3C drafts cited (see Resources). In addition, there are other closely associated areas, such as authenticating users or systems, identifying levels of authorisation, and managing keys, that are all relevant to XML security.

SAML is an imitative driven by OASIS that attempts to blend the competing specifications AuthML and S2ML, and to facilitate the exchange of authentication and authorisation information. Closely related to SAML, but focusing more on a subject-privilege-object orientated security model in the context of a particular XML document, is the eXtensible Access Control Markup Language, also directed by OASIS and variously known (even within the same documents) as XACML or XACL. By writing rules in XACL, a policy author can define who can exercise what access privileges for a particular XML document, something relevant in the situations cited earlier.

XKMS, now being considered by a W3C committee, is intended to establish a protocol for key management on top of the XML signature standard. With SAML, XACL, and other initiatives, XKMS is an important element in the large jigsaw that makes up security as applied to XML documents. Its immediate effect is to simplify greatly the management of authentication and signature keys; it does this by separating the function of digital certificate processing, revocation status checking, and certification path location and validation from the application involved -- for example, by delegating key management to an Internet Web service.

XML security still has a long way to go before the needs of convenience in use, reliability, and robustness are met. But good progress is being made.

Resources

- Digital signatures for SOAP messages, a developerWorks tutorial by Jayanthi Suryanarayana, explains how to digitally sign and encrypt your SOAP messages for security.
- The XML Security Suite: Increasing the security of e-business by Doug Tidwell presents some basics of Web security, describes the components of the XML Security Suite, and gives examples that illustrate how the technologies in the XML Security Suite increase the security of Web commerce.
- The OASIS consortium site includes The XML Cover Pages: XML and Encryption, Robin Cover' active diary of activities and publications relevant to these issues. The site also features a draft document specifying the Security Assertion Markup Language (SAML).
- The W3C working draft XML Encryption Requirements lists the design principles, scope, and requirements for the XML Encryption. It includes requirements as they relate to the encryption syntax, data model, format, cryptographic processing, and external requirements and coordination.
- XML Encryption Syntax and Processing specifies a process for encrypting data and representing the result in XML. The data may be arbitrary data (including an XML document), an XML element, or XML element content.
- XML-Signature Requirements lists the design principles, scope, and requirements for the XML Digital Signature specification. It includes requirements as they relate to the signature syntax, data model, format, cryptographic processing, and external requirements and coordination.
- XML-Signature Syntax and Processing specifies XML digital signature processing rules and syntax. XML signatures provide integrity, message authentication, and/or signer authentication services for data of any type, whether located

within the XML that includes the signature or elsewhere.

- Decryption Transform for XML signature specifies the "decryption transform," which enables XML signatures verification even if both signature and encryption operations are performed on an XML document.
- XML Key Management Specification specifies protocols for distributing and registering public keys, suitable for use in conjunction with the proposed standard for XML signature [XML-SIG] developed by the W3C and the Internet Engineering Task Force (IETF) and an anticipated companion standard for XML encryption.
- Find out the latest security-related news and product information at IBM's security site.
- Get an overview of the IBM WebSphere 3.x server security model.

About the author

Murdoch Mactaggart is a freelance writer and business consultant who writes on software development, security and cryptography, the Internet, and on business and management issues around these areas. He first bought a computer in 1979 to run what was then his business as a dealer in antiquarian books and Old Master and Japanese prints. He has programmed extensively using Assembler, Pascal, MegaBasic, Icon, Revelation, C, VB, C++, and various Web tools. Ever since writing a text processor in Z80 machine code he's felt there had to be more to life and is, in consequence, a strong advocate of interoperability and of using mechanisms to minimise repetitive coding. Contact him at IBMDev@TextBiz.com.

e-mail it!

What do you think of this article?

○ Killer! (5) ○ Good stuff (4) ○ So-so; not bad (3) ○ Needs work (2) ○ Lame! (1)

Comments?

Submit feedback

About IBM | Privacy | Legal | Contact