XML in the Development of Component Systems

XML Signature and Encryption

Overview

- Canonicalization
- Signature
- Encryption

Canonical XML (1)

- http://www.w3.org/TR/2001/REC-xml-c14n-20010315
- Definition of canonical form:
  - Document is encoded in UTF-8
  - Line breaks are normalized as #A
  - Attribute values are normalized as if done by normalizing parser
  - Character and parsed entity references are replaced with their character content
  - Empty elements are converted to start-end pair tags
  - Whitespace outside the document element and within tags is normalized
  - Attribute value delimiters are set to quotation marks (double quotes)
  - Special characters in attributes and character content are replaced by character references
  - Superfluous namespace definitions are removed
  - Default attributes are added to each element
  - Lexicographic order is imposed on the namespace declarations and attributes of each element (sorted by namespace URI)

Canonical XML (2)

- Input to canonicalization is an XPath node set
  - Input is converted to a node set if it is a byte stream
  - optional elimination of XML comments
- Conversion to node set
  - CDATA sections are eliminated
  - Conversion to NFC if the input is not UTF-*
- Output starts with root node
  - no byte order mark, XML declaration, or DOCTYPE declaration
- Element nodes: namespaces first, then attributes, then child nodes
  - Optionally emit empty namespace (atmost once)
  - eliminate namespaces with identical prefix and URI as in parent node
  - attributes: space, attribute name, =, “, value, “; escape &<"	

Canonical XML(3)

- No XML declaration
  - version is defined to be 1.0, encoding is UTF-8
- Whitespace outside document element
  - normally eliminated, but PIs are preceded/followed by

- No namespace prefix rewriting
  - would break documents that use namespace prefixes in attribute values/character data

Canonical XML (4)

- Canonicalization can be applied to a subset of the document
  - by default, entire document is normalized
    - without comments (//. | //@* | //namespace::*)[not(self::comment())]
    - with comments (//. | //@* | //namespace::*)
- For a subset, the parent node might not be included in the output
  - in this case, the node inherits all attributes with the xml: prefix

Exclusive Canonicalization

- http://www.w3.org/TR/2002/REC-xml-exc-c14n-20020718/
- Namespace declaration are only included in a document subset if the subset “visibly utilizes” the namespace declaration
  - an element or an attribute names uses the prefix
- Does not include xml:lang, xml:base, and xml:space in child nodes

XML Signature

- http://www.w3.org/TR/xmldsig-core/
  XML-Signature Syntax and Processing
- Signatures are XML documents
- integrity, message authentication and/or signer authentication
  - for data of any kind (in the XML document or outside)
- enveloping signature
  - data is enclosed in signature
- enveloped signature
  - signature is enclosed in the data
- detached signature
  - signature is external to the object (referred to by URI)

Signature Schema

- Namespace http://www.w3.org/2000/09/xmldsig#
  - Typically used with prefix “dsig”
  - URI is also a prefix for algorithm identifiers
- Signature elements defined in Schema and DTD
  - Schema definition is normative

Signature Overview

- digital content (data objects) is digested (hashed)
- hash value is cryptographically signed

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

Signature Example (1/2)

```
<Signature Id="MyFirstSignature" xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
   <CanonicalizationMethod
    Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
   <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
   <Reference URI="http://www.w3.org/TR/2000/REC-xhtml1-20000126/">
    <Transforms>
     <Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-
    c14n-20010315"/>
    </Transforms>
    <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
    <DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
  </Reference>
  </SignedInfo>
…
```

Signature Example (2/2)

```
  <SignatureValue>MC0CFFrVLtRlk=...</SignatureValue>
  <KeyInfo>
    <KeyValue>
      <DSAKeyValue>
        <P>...</P><Q>...</Q><G>...</G><Y>...</Y>
      </DSAKeyValue>
    </KeyValue>
  </KeyInfo>
</Signature>
```

Signature Properties

- Additional signed information in the signature

```
<Signature Id="MySecondSignature" ...>…
<Reference URI="http://www.w3.org/TR/xml-stylesheet/">
<Reference URI="#AMadeUpTimeStamp"
        Type="http://www.w3.org/2000/09/xmldsig#SignatureProperties">
  <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
  <DigestValue>k3453rvEPO0vKtMup4NbeVu8nk=</DigestValue>
</Reference>
...
<Object>
 <SignatureProperties>
   <SignatureProperty Id="AMadeUpTimeStamp" Target="#MySecondSignature">
   <timestamp xmlns="http://www.ietf.org/rfcXXXX.txt">
    <date>19990908</date>
    <time>14:34:34:34</time>
   </timestamp>
  </SignatureProperty>
 </SignatureProperties>
</Object>
```

Manifests

- Collections of references, for use in multiple signatures

```
<Reference URI="#MyFirstManifest"
  Type="http://www.w3.org/2000/09/xmldsig#Manifest">
 <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
 <DigestValue>345x3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
</Reference>

<Object>
 <Manifest Id="MyFirstManifest">
  <Reference> ... </Reference>
  <Reference> ... </Reference>
  </Manifest>
</Object>
```

Core Generation

- Algorithm to follow for generating the signature:

1. Reference Generation
   1. Apply the transforms
   2. Calculate the digest
   3. Create the Reference element, including identification of the data object, any transforms, the digest algorithm and the digest value
2. Signature Generation
   1. Create SignedInfo (SignatureMethod, CanonicalizationMethod, References)
   2. Canonicalize SignatureInfo, and compute signature method
   3. Construct Signature (SignedInfo, Objects, KeyInfo, SignatureValue)

Core Validation

- Reference validation and core validation both required

1. Canonicalize the SignedInfo
2. Reference Validation
   1. Obtain data to be digested
   2. Digest the resulting data using the DigestMethod
   3. Compare the generated value against the DigestValue
3. Signature Validation
   1. Obtain key information (from KeyInfo or external source)
   2. Apply CanonicalizationMethod and SignatureMethod for SignedInfo, confirm SignatureValue

CryptoBinary

- Data type to represent arbitrarily large numbers in binary
  1. Convert number to a bit string
  2. pad with leading zero bits to obtain an integral number of bytes
  3. represent byte string as big-endian base64

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

Canonicalization

- Required Algorithms:
  - Canonical XML (omits comments): http://www.w3.org/TR/2001/REC-xml-c14n-20010315
  - Canonical XML (with comments): http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments

Signature

- Required Algorithms
  - DSAwithSHA1 (DSS) http://www.w3.org/2000/09/xmldsig#dsa-sha1
  - HMAC http://www.w3.org/2000/09/xmldsig#hmac-sha1
- Recommended Algorithms
  - RSAwithSHA1 http://www.w3.org/2000/09/xmldsig#rsa-sha1

Reference

- URI specifies document to sign
  - Fragment identifier may refer to Id in the same document, based on XPointer
- Transformation Algorithms:
  - Optional XSLT http://www.w3.org/2000/09/xmldsig#rsa-sha1
  - Recommended XPath http://www.w3.org/TR/1999/REC-xpath-19991116
  - Required Enveloped Signature http://www.w3.org/2000/09/xmldsig#enveloped-signature
    - Defined as an XPath expression that removes the signature from the document
- Default transformation: C14N

XPath Transform

- Input: node set (//. | //@* | //namespace::*)
- Output: node set that is meant to be signed
- Include all nodes for which the expression evaluates to true

```
<Reference URI="">
    <Transforms>
     <Transform Algorithm="http://www.w3.org/TR/1999/REC-xpath-19991116">
       <XPath xmlns:dsig="http://www.w3.org/2000/09/xmldsig# ">
       not(ancestor-or-self::dsig:Signature)
       </XPath>
     </Transform>
    </Transforms>
    <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
    <DigestValue></DigestValue>
</Reference>
```

- Additional function here() refers to the XPath element

XSLT Transform

- Child of Transform element is xsl:stylesheet
- XSLT requires an octet stream as input
  - perform C14N first if actual input is a node set
- Output method should be XML, signature authors should apply C14N on output, since actual output is underspecified

Digest

- Required SHA1 http://www.w3.org/2000/09/xmldsig#sha1

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"/>
      <element ref="ds:PGPData"/>
      <element ref="ds:SPKIData"/>
      <element ref="ds:MgmtData"/>
      <any processContents="lax" namespace="##other"/>
      <!-- (1,1) elements from (0,unbounded) namespaces -->
    </choice>
    <attribute name="Id" type="ID" use="optional"/>
  </complexType>
```

XML Encryption

- XML Encryption Syntax and Processing
  http://www.w3.org/TR/xmlenc-core/
- xmlns:xenc='http://www.w3.org/2001/04/xmlenc#‘
  - namespace also serves as a prefix for algorithm identifiers
- Encryption granularity:
  - document
  - element
  - element content
  - non-XML data
- Allows additional transmission of encrypted keys
  - CarriedKeyName allows to refer to the key

Encryption Syntax Overview

```
<EncryptedData Id? Type? MimeType? Encoding?>
  <EncryptionMethod/>?
  <ds:KeyInfo>
    <EncryptedKey>?
    <AgreementMethod>?
    <ds:KeyName>?
    <ds:RetrievalMethod>?
    <ds:*>?
  </ds:KeyInfo>?
  <CipherData>
    <CipherValue>?
    <CipherReference URI?>?
  </CipherData>
  <EncryptionProperties>?
 </EncryptedData>
```

Encryption Example: Plaintext

```
<?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>Example Bank</Issuer>
      <Expiration>04/02</Expiration>
    </CreditCard>
  </PaymentInfo>
```

Encryption Example: Element

```
<?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>
```

Encryption Example: Content

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

Encryption Example: Non-XML

```
<?xml version='1.0'?>
  <EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
   MimeType=‘image/gif'>
    <CipherData>
      <CipherValue>A23B45C56</CipherValue>
    </CipherData>
  </EncryptedData>
```

EncryptedKey

- Symmetric Key: KeyName

```
<EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
    Type='http://www.w3.org/2001/04/xmlenc#Element'/>
  <EncryptionMethod
    Algorithm='http://www.w3.org/2001/04/xmlenc#tripledes-cbc'/>
  <ds:KeyInfo xmlns:ds='http://www.w3.org/2000/09/xmldsig#'>
   <ds:KeyName>John Smith</ds:KeyName>
  </ds:KeyInfo>
  <CipherData><CipherValue>DEADBEEF</CipherValue></
   CipherData>
 </EncryptedData>
```

- Alternatively: use ds:RetrievalMethod to fetch the key

CipherData

- Base-64 data in CipherValue
- Alternatively, CipherReference refers to external encrypted data

```
<CipherReference URI="http://www.example.com/CipherValues.xml">
  <Transforms>
    <ds:Transform
     Algorithm="http://www.w3.org/TR/1999/REC-xpath-19991116">
       <ds:XPath xmlns:rep="http://www.example.org/repository">
         self::text()[parent::rep:CipherValue[@Id="example1"]]
       </ds:XPath>
    </ds:Transform>
    <ds:Transform Algorithm="http://www.w3.org/2000/09/
   xmldsig#base64"/>
  </Transforms>
 </CipherReference>
```

Encryption Processing

1. Select the algorithm
2. Obtain (and optionally represent) the key
   - put ds:KeyName, ds:KeyValue, ds:RetrievalMethod, xenc:EncryptedKey into ds:KeyInfo as desired
3. Encrypt the data
   1. For XML element or content, build UTF-8 bytes
   2. Otherwise, use resource octets
   3. Optionally, include MIME type in encrypted data
4. Build EncryptedType (EncryptedData or EncryptedKey)
5. Put encryption results into target document

Decryption Processing

- For each EncryptedData or EncryptedKey
  1. Determine algorithm and key information
     - Application must supply missing information (e.g. private keys)
  2. Locate the key needed to decrypt, either from the application, from the same document (EncryptedKey), or through the RetrievalMethod
  3. Decrypt the CipherData
     - For CipherReference, perform download first
  4. Parse data of type Element or Content
     - Recursively attempt decrypting embedded encrypted data
  5. Process non-XML data appropriately

Encryption Algorithms

- Block Encryption
  - Required TRIPLEDES http://www.w3.org/2001/04/xmlenc#tripledes-cbc
  - Required AES-128 http://www.w3.org/2001/04/xmlenc#aes128-cbc
  - Required AES-256 http://www.w3.org/2001/04/xmlenc#aes256-cbc
  - Optional AES-192 http://www.w3.org/2001/04/xmlenc#aes192-cbc
- Stream Encryption: none
- Key Transport
  - Required RSA-v1.5 http://www.w3.org/2001/04/xmlenc#rsa-1_5
  - Required RSA-OAEP http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p
- Key Agreement
  - Optional Diffie-Hellman http://www.w3.org/2001/04/xmlenc#dh

Encryption Algorithms (2)

- Symmetric Key Wrap
  - Required TRIPLEDES KeyWrap http://www.w3.org/2001/04/xmlenc#kw-tripledes
  - Required AES-128 KeyWrap http://www.w3.org/2001/04/xmlenc#kw-aes128
  - Required AES-256 KeyWrap http://www.w3.org/2001/04/xmlenc#kw-aes256
  - Optional AES-192 KeyWrap http://www.w3.org/2001/04/xmlenc#kw-aes192
- Message Digest
  - Required SHA1 http://www.w3.org/2000/09/xmldsig#sha1
  - Recommended SHA256 http://www.w3.org/2001/04/xmlenc#sha256
  - Optional SHA512 http://www.w3.org/2001/04/xmlenc#sha512
  - Optional RIPEMD-160 http://www.w3.org/2001/04/xmlenc#ripemd160

Encryption Algorithms (3)

- Message Authentication
  - Recommended XML Digital Signature http://www.w3.org/2000/09/xmldsig#
- Canonicalization
  - Optional Canonical XML (omits comments) http://www.w3.org/TR/2001/REC-xml-c14n-20010315
  - Optional Canonical XML with comments http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments
  - Optional Exclusive XML Canonicalization http://www.w3.org/2001/10/xml-exc-c14n#
  - Optional Exclusive XML Canonicalization with comments http://www.w3.org/2001/10/xml-exc-c14n#WithComments
- Encoding
  - Required base64 http://www.w3.org/2000/09/xmldsig#base64

Summary

- Involved specifications
  - flexibility in underlying algorithms
  - flexibility in integration into XML
- Not stand-alone, but embedded in other applications
  - WS-Security
  - Elster Online