Additional XML Security Uniform Resource Identifiers (URIs)

Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2005).

Abstract

A number of Uniform Resource Identifiers (URIs) intended for use with XML Digital Signatures, Encryption, and Canonicalization are defined. These URIs identify algorithms and types of keying information.

Table of Contents

1.  Introduction.................................................. 2
2.  Algorithms.................................................... 3
   2.1.  DigestMethod Algorithms................................ 3
      2.1.1.  MD5............................................. 3
      2.1.2.  SHA-224........................................ 3
      2.1.3.  SHA-384........................................ 4
   2.2.  SignatureMethod Message Authentication Code Algorithms.. 4
      2.2.1.  HMAC-MD5........................................ 4
      2.2.2.  HMAC SHA Variations............................ 5
      2.2.3.  HMAC-RIPEMD160................................. 6
   2.3.  SignatureMethod Public Key Signature Algorithms........ 6
      2.3.1.  RSA-MD5........................................ 6
      2.3.2.  RSA-SHA256..................................... 7
      2.3.3.  RSA-SHA384..................................... 7
      2.3.4.  RSA-SHA512..................................... 7
      2.3.5.  RSA-RIPEMD160.................................. 8
      2.3.6.  ECDSA-SHA*..................................... 8
      2.3.7.  ESIGN-SHA1..................................... 8
   2.4.  Minimal Canonicalization................................ 9
   2.5.  Transform Algorithms.................................... 9
      2.5.1.  XPointer....................................... 9
1. Introduction

XML Digital Signatures, Canonicalization, and Encryption have been standardized by the W3C and the joint IETF/W3C XMLDSIG working group. All of these are now W3C Recommendations and IETF Informational or Standards Track documents. They are available as follows:

<table>
<thead>
<tr>
<th>IETF level</th>
<th>W3C REC</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>[RFC3076]</td>
<td>Info [CANON]</td>
<td>Canonical XML</td>
</tr>
<tr>
<td>- - - - -</td>
<td>[XMLENC]</td>
<td>XML Encryption</td>
</tr>
<tr>
<td>[RFC3741]</td>
<td>Info [EXCANON]</td>
<td>Exclusive XML Canonicalization</td>
</tr>
</tbody>
</table>

All of these standards and recommendations use URIs [RFC2396] to identify algorithms and keying information types. This document provides a convenient reference list of URIs and descriptions for algorithms in which there is substantial interest, but which cannot or have not been included in the main documents. Note that raising XML digital signature to a Draft Standard in the IETF required removal of any algorithms for which interoperability from the main standards document has not been demonstrated. This required removal of the Minimal Canonicalization algorithm, in which there appears to be a continued interest, to be dropped from the standards track specification. It is included here.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].
2. Algorithms

The URI [RFC2396] being dropped from the standard because of the transition from Proposed Standard to Draft Standard is included in Section 2.4 with its original prefix so as to avoid changing the XMLDSIG standard’s namespace.

http://www.w3.org/2000/09/xmldsig#

Additional algorithms are given URIs that start with:

http://www.w3.org/2001/04/xmldsig-more#

An "xmldsig-more" URI does not imply any official W3C status for these algorithms or identifiers or that they are only useful in digital signatures. Currently, dereferencing such URIs may or may not produce a temporary placeholder document. Permission to use this URI prefix has been given by the W3C.

2.1. DigestMethod Algorithms

These algorithms are usable wherever a DigestMethod element occurs.

2.1.1. MD5

Identifier:

http://www.w3.org/2001/04/xmldsig-more#md5

The MD5 algorithm [RFC1321] takes no explicit parameters. An example of an MD5 DigestAlgorithm element is:

<DigestAlgorithm
  Algorithm="http://www.w3.org/2001/04/xmldsig-more#md5"/>

An MD5 digest is a 128-bit string. The content of the DigestValue element shall be the base64 [RFC2405] encoding of this bit string viewed as a 16-octet octet stream.

2.1.2. SHA-224

Identifier:

http://www.w3.org/2001/04/xmldsig-more#sha224

The SHA-224 algorithm [FIPS-180-2change, RFC3874] takes no explicit parameters. An example of a SHA-224 DigestAlgorithm element is:
<DigestAlgorithm
   Algorithm="http://www.w3.org/2001/04/xmlsig-more#sha224" />

A SHA-224 digest is a 224 bit string. The content of the DigestValue element shall be the base64 [RFC2405] encoding of this string viewed as a 28-octet stream. Because it takes roughly the same amount of effort to compute a SHA-224 message digest as a SHA-256 digest, and terseness is usually not a criteria in an XML application, consideration should be given to the use of SHA-256 as an alternative.

2.1.3. SHA-384

Identifier:
   http://www.w3.org/2001/04/xmlsig-more#sha384

The SHA-384 algorithm [FIPS-180-2] takes no explicit parameters. An example of a SHA-384 DigestAlgorithm element is:

<DigestAlgorithm
   Algorithm="http://www.w3.org/2001/04/xmlsig-more#sha384" />

A SHA-384 digest is a 384 bit string. The content of the DigestValue element shall be the base64 [RFC2405] encoding of this string viewed as a 48-octet stream. Because it takes roughly the same amount of effort to compute a SHA-384 message digest as a SHA-512 digest and terseness is usually not a criteria in XML application, consideration should be given to the use of SHA-512 as an alternative.

2.2. SignatureMethod Message Authentication Code Algorithms

   Note: Some text in this section is duplicated from [RFC3275] for the convenience of the reader. RFC 3275 is normative in case of conflict.

2.2.1. HMAC-MD5

Identifier:
   http://www.w3.org/2001/04/xmlsig-more#hmac-md5

The HMAC algorithm [RFC2104] takes the truncation length in bits as a parameter; if the parameter is not specified then all the bits of the hash are output. An example of an HMAC-MD5 SignatureMethod element is as follows:
<SignatureMethod>
  Algorithm="http://www.w3.org/2001/04/xmldsig-more#hmac-md5">
  <HMACOutputLength>112</HMACOutputLength>
</SignatureMethod>

The output of the HMAC algorithm is ultimately the output (possibly truncated) of the chosen digest algorithm. This value shall be base64 [RFC2405] encoded in the same straightforward fashion as the output of the digest algorithms. For example, the SignatureValue element for the HMAC-MD5 digest

9294727A 3638BB1C 13F48EF8 158BFC9D

from the test vectors in [RFC2104] would be

kpRyejY4uxwT9I74FYv8nQ==

Schema Definition:

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

DTD:

<!ELEMENT HMACOutputLength (#PCDATA) >

The Schema Definition and DTD immediately shown above are taken from [RFC3275].

Although some cryptographic suspicions have recently been cast on MD5 for use in signatures such as RSA-MD5 below, this does not effect use of MD5 in HMAC.

2.2.2. HMAC SHA Variations

Identifiers:

http://www.w3.org/2001/04/xmldsig-more#hmac-sha224
http://www.w3.org/2001/04/xmldsig-more#hmac-sha256
http://www.w3.org/2001/04/xmldsig-more#hmac-sha384
http://www.w3.org/2001/04/xmldsig-more#hmac-sha512

SHA-224, SHA-256, SHA-384, and SHA-512 [FIPS-180-2, FIPS-180-2change, RFC3874] can also be used in HMAC as described in section 2.2.1 for HMAC-MD5.
2.2.3. HMAC-RIPEMD160

Identifier:  
http://www.w3.org/2001/04/xmldsig-more#hmac-ripemd160

RIPEMD-160 [RIPEMD-160] can also be used in HMAC as described in section 2.2.1 for HMAC-MD5.

2.3. SignatureMethod Public Key Signature Algorithms

These algorithms are distinguished from those in Section 2.2 in that they use public key methods. The verification key is different from and not feasibly derivable from the signing key.

2.3.1. RSA-MD5

Identifier:  
http://www.w3.org/2001/04/xmldsig-more#rsa-md5

RSA-MD5 implies the PKCS#1 v1.5 padding algorithm described in [RFC3447]. An example of use is

<SignatureMethod  
Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-md5" />

The SignatureValue content for an RSA-MD5 signature is the base64 [RFC2405] encoding of the octet string computed as per [RFC3447], section 8.1.1, signature generation for the RSASSA-PKCS1-v1_5 signature scheme. As specified in the EMSA-PKCS1-V1_5-ENCODER function in [RFC3447, section 9.2.1], the value input to the signature function MUST contain a pre-pended algorithm object identifier for the hash function, but the availability of an ASN.1 parser and recognition of OIDs are not required of a signature verifier. The PKCS#1 v1.5 representation appears as:

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

Note that the padded ASN.1 will be of the following form:

01 | FF* | 00 | prefix | hash

Vertical bar ("|") represents concatenation. "01", "FF", and "00" are fixed octets of the corresponding hexadecimal value and the asterisk ("*") after "FF" indicates repetition. "hash" is the MD5 digest of the data. "prefix" is the ASN.1 BER MD5 algorithm designator prefix required in PKCS #1 [RFC3447], that is:

hex 30 20 30 0c 06 08 2a 86 48 86 f7 0d 02 05 00 04 10
This prefix is included to facilitate the use of standard cryptographic libraries. The FF octet MUST be repeated enough times that the value of the quantity being CRYPTed is exactly one octet shorter than the RSA modulus.

Due to increases in computer processor power and advances in cryptography, use of RSA-MD5 is NOT RECOMMENDED.

### 2.3.2. RSA-SHA256

Identifier:  
http://www.w3.org/2001/04/xmldsig-more#rsa-sha256

This implies the PKCS#1 v1.5 padding algorithm [RFC3447] as described in section 2.3.1, but with the ASN.1 BER SHA-256 algorithm designator prefix. An example of use is:

```xml
<SignatureMethod
    Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256" />
```

### 2.3.3 RSA-SHA384

Identifier:  
http://www.w3.org/2001/04/xmldsig-more#rsa-sha384

This implies the PKCS#1 v1.5 padding algorithm [RFC3447] as described in section 2.3.1, but with the ASN.1 BER SHA-384 algorithm designator prefix. An example of use is:

```xml
<SignatureMethod
    Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha384" />
```

Because it takes about the same effort to calculate a SHA-384 message digest as a SHA-512 message digest, it is suggested that RSA-SHA512 be used in preference to RSA-SHA384 where possible.

### 2.3.4. RSA-SHA512

Identifier:  
http://www.w3.org/2001/04/xmldsig-more#rsa-sha512

This implies the PKCS#1 v1.5 padding algorithm [RFC3447] as described in section 2.3.1, but with the ASN.1 BER SHA-512 algorithm designator prefix. An example of use is:

```xml
<SignatureMethod
    Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha512" />
```
2.3.5. RSA-RIPEMD160

Identifier:
http://www.w3.org/2001/04/xmldsig-more/rsa-ripemd160

This implies the PKCS#1 v1.5 padding algorithm [RFC3447], as described in section 2.3.1, but with the ASN.1 BER RIPEMD160 algorithm designator prefix. An example of use is:

<SignatureMethod
  Algorithm="http://www.w3.org/2001/04/xmldsig-more/rsa-ripemd160" />

2.3.6. ECDSA-SHA*

Identifiers
  http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha1
  http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha224
  http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha256
  http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha384
  http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha512

The Elliptic Curve Digital Signature Algorithm (ECDSA) [FIPS-186-2] is the elliptic curve analogue of the DSA (DSS) signature method. For detailed specifications on how to use it with SHA hash functions and XML Digital Signature, please see [X9.62] and [ECDSA].

2.3.7. ESIGN-SHA1

Identifier
  http://www.w3.org/2001/04/xmldsig-more#esign-sha1
  http://www.w3.org/2001/04/xmldsig-more#esign-sha224
  http://www.w3.org/2001/04/xmldsig-more#esign-sha256
  http://www.w3.org/2001/04/xmldsig-more#esign-sha384
  http://www.w3.org/2001/04/xmldsig-more#esign-sha512

The ESIGN algorithm specified in [IEEE-P1363a] is a signature scheme based on the integer factorization problem. It is much faster than previous digital signature schemes so ESIGN can be implemented on smart cards without special co-processors.

An example of use is:

<SignatureMethod
  Algorithm="http://www.w3.org/2001/04/xmldsig-more#esign-sha1" />
2.4. Minimal Canonicalization

Thus far two independent interoperable implementations of Minimal Canonicalization have not been announced. Therefore, when XML Digital Signature was advanced from Proposed Standard [RFC3075] to Draft Standard [RFC3275], Minimal Canonicalization was dropped from the standards track documents. However, there is still interest in Minimal Canonicalization, indicating its possible future use. For its definition, see [RFC3075], Section 6.5.1.

For reference, its identifier remains:
http://www.w3.org/2000/09/xmldsig#minimal

2.5. Transform Algorithms

Note that all CanonicalizationMethod algorithms can also be used as transform algorithms.

2.5.1. XPointer

Identifier:
http://www.w3.org/2001/04/xmldsig-more/xptr

This transform algorithm takes an [XPointer] as an explicit parameter. An example of use is [RFC3092]:

```xml
<Transform
  Algorithm='http://www.w3.org/2001/04/xmldsig-more/xptr'>
  <XPointer
    xmlns='http://www.w3.org/2001/04/xmldsig-more/xptr'>
    xpointer(id("foo"))
    xmlns(bar='http://foobar.example')
    xpointer(//bar:Zab[@Id="foo"])  
  </XPointer>
</Transform>
```

Schema Definition:

```xml
<element name="XPointer" type="string">
  <!ELEMENT XPointer (#PCDATA) >
```

Input to this transform is an octet stream (which is then parsed into XML).
Output from this transform is a node set; the results of the XPointer are processed as defined in the XMLDSIG specification [RFC3275] for a same document XPointer.

2.6. EncryptionMethod Algorithms

This subsection gives identifiers and information for several EncryptionMethod Algorithms.

2.6.1. ARCFOUR Encryption Algorithm

Identifier:

http://www.w3.org/2001/04/xmldsig-more#arcfour

ARCFOUR is a fast, simple stream encryption algorithm that is compatible with RSA Security’s RC4 algorithm. An example of the EncryptionMethod element using ARCFOUR is

```xml
<EncryptionMethod
   Algorithm="http://www.w3.org/2001/04/xmldsig-more#arcfour">
   <KeySize>40</KeySize>
</EncryptionMethod>
```

Note that Arcfour makes use of the generic KeySize parameter specified and defined in [XMLENC].

2.6.2. Camellia Block Encryption

Identifiers:

http://www.w3.org/2001/04/xmldsig-more#camellia128-cbc
http://www.w3.org/2001/04/xmldsig-more#camellia192-cbc
http://www.w3.org/2001/04/xmldsig-more#camellia256-cbc

Camellia is an efficient and secure block cipher with the same interface as the AES [Camellia, RFC3713], that is 128-bit block size and 128, 192, and 256 bit key sizes. In XML Encryption, Camellia is used in the same way as the AES: It is used in the Cipher Block Chaining (CBC) mode with a 128-bit initialization vector (IV). The resulting cipher text is prefixed by the IV. If included in XML output, it is then base64 encoded. An example Camellia EncryptionMethod is as follows:

```xml
<EncryptionMethod
   Algorithm="http://www.w3.org/2001/04/xmldsig-more#camellia128-cbc" />
```
2.6.3. Camellia Key Wrap

Identifiers:

http://www.w3.org/2001/04/xmldsig-more#kw-camellia128
http://www.w3.org/2001/04/xmldsig-more#kw-camellia192
http://www.w3.org/2001/04/xmldsig-more#kw-camellia256

The Camellia [Camellia, RFC3713] key wrap is identical to the AES key wrap algorithm [RFC3394] specified in the XML Encryption standard with "AES" replaced by "Camellia". As with AES key wrap, the check value is 0xA6A6A6A6A6A6A6A6.

The algorithm is the same regardless of the size of the Camellia key used in wrapping (called the key encrypting key or KEK). The implementation of Camellia is OPTIONAL. However, if it is supported, the same implementation guidelines of which combinations of KEK size and wrapped key size should be required to be supported and which are optional to be supported should be followed as for AES. That is to say, if Camellia key wrap is supported, then wrapping 128-bit keys with a 128-bit KEK and wrapping 256-bit keys with a 256-bit KEK are REQUIRED and all other combinations are OPTIONAL.

An example of use is:

```xml
<EncryptionMethod
    Algorithm="http://www.w3.org/2001/04/xmldsig-more#kw-camellia128" />
```

2.6.4. PSEC-KEM

Identifier:

http://www.w3.org/2001/04/xmldsig-more#psec-kem

The PSEC-KEM algorithm, specified in [ISO/IEC-18033-2], is a key encapsulation mechanism using elliptic curve encryption.

An example of use is:

```xml
<EncryptionMethod
    Algorithm="http://www.w3.org/2001/04/xmlenc#psec-kem">
    <ECPARAMETERS>
        <Version>version</Version>
        <FieldID>id</FieldID>
        <Curve>curve</Curve>
        <Base>base</Base>
        <Order>order</Order>
        <Cofactor>cofactor</Cofactor>
    </ECPARAMETERS>
```

3. KeyInfo

In section 3.1 a new KeyInfo element child is specified, while in section 3.2 additional KeyInfo Type values for use in RetrievalMethod are specified.

3.1. PKCS #7 Bag of Certificates and CRLs

A PKCS #7 [RFC2315] "signedData" can also be used as a bag of certificates and/or certificate revocation lists (CRLs). The PKCS7signedData element is defined to accommodate such structures within KeyInfo. The binary PKCS #7 structure is base64 [RFC2405] encoded. Any signer information present is ignored. The following is an example, eliding the base64 data [RFC3092]:

```xml
<foo:PKCS7signedData
   xmlns:foo="http://www.w3.org/2001/04/xmldsig-more">
   ...
</foo:PKCS7signedData>
```

3.2. Additional RetrievalMethod Type Values

The Type attribute of RetrievalMethod is an optional identifier for the type of data to be retrieved. The result of dereferencing a RetrievalMethod reference for all KeyInfo types with an XML structure is an XML element or document with that element as the root. The various "raw" key information types return a binary value. Thus, they require a Type attribute because they are not unambiguously parseable.

Identifiers:

- http://www.w3.org/2001/04/xmldsig-more#KeyValue
- http://www.w3.org/2001/04/xmldsig-more#RetrievalMethod
- http://www.w3.org/2001/04/xmldsig-more#KeyName
- http://www.w3.org/2001/04/xmldsig-more#rawX509CRL
- http://www.w3.org/2001/04/xmldsig-more#rawPGPKeyPacket
- http://www.w3.org/2001/04/xmldsig-more#rawSPKISexp
- http://www.w3.org/2001/04/xmldsig-more#PKCS7signedData
- http://www.w3.org/2001/04/xmldsig-more#rawPKCS7signedData
4. IANA Considerations

As it is easy for people to construct their own unique URIs [RFC2396] and possibly obtain a URI from the W3C if appropriate, it is not intended that any additional "http://www.w3.org/2001/04/xmlsig-more#" URIs be created beyond those enumerated in this document. (W3C Namespace stability rules prohibit the creation of new URIs under "http://www.w3.org/2000/09/xmlsig#").

5. Security Considerations

Due to computer speed and cryptographic advances, the use of MD5 as a DigestMethod and the use of MD5 in the RSA-MD5 SignatureMethod is NOT RECOMMENDED. The concerned cryptographic advances do not effect the security of HMAC-MD5; however, there is little reason not to use one of the SHA series of algorithms.

Acknowledgements

Glenn Adams, Merlin Hughs, Gregor Karlinger, Brian LaMachia, Shiho Moriai, Joseph Reagle, Russ Housley, and Joel Halpern.

Normative References


[RFC3075] Eastlake 3rd, D., Reagle, J., and D. Solo, "XML-Signature Syntax and Processing", RFC 3075, March 2001. (RFC 3075 was obsoleted by RFC 3275 but is referenced in this document for its description of Minimal Canonicalization which was dropped in RFC 3275.)


Informative References


Author’s Address

Donald E. Eastlake 3rd
Motorola Laboratories
155 Beaver Street
Milford, MA 01757 USA

Phone: +1-508-786-7554 (w)
       +1-508-634-2066 (h)
EMail: Donald.Eastlake@motorola.com
Full Copyright Statement

Copyright (C) The Internet Society (2005).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.