Internet Engineering Task Force (IETF)
Request for Comments: 6931
Obsoletes: 4051
Category: Standards Track
ISSN: 2070-1721

D. Eastlake 3rd
Huawei
April 2013

Additional XML Security Uniform Resource Identifiers (URIs)

Abstract

This document expands, updates, and establishes an IANA registry for the list of URIs intended for use with XML digital signatures, encryption, canonicalization, and key management. These URIs identify algorithms and types of information. This document obsoletes RFC 4051.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6931.

Copyright Notice

Table of Contents

1. Introduction

XML digital signatures, canonicalization, and encryption have been standardized by the W3C and by the joint IETF/W3C XMLDSIG working group [W3C]. All of these are now W3C Recommendations and some are also RFCs. They are available as follows:

RFC Status	W3C REC	Topic
[RFC3275] Draft Standard	[XMLDSIG10]	XML Digital Signatures
[RFC3076] Informational	[CANON10]	Canonical XML
- - - - - -	[XMLENC10]	XML Encryption 1.0
[RFC3741] Informational	[XCANON]	Exclusive XML Canonicalization 1.0

All of these documents and recommendations use URIs [RFC3986] to identify algorithms and keying information types. The W3C has subsequently produced updated XML Signature 1.1 [XMLDSIG11], Canonical XML 1.1 [CANON11], and XML Encryption 1.1 [XMLENC11] versions, as well as a new XML Signature Properties specification [XMLDSIG-PROP].

All camel-case element names herein, such as DigestValue, are from these documents.

This document is an updated convenient reference list of URIs and corresponding algorithms in which there is expressed interest. Since the previous list [RFC4051] was issued in 2005, significant new cryptographic algorithms of interest to XML security, for some of which the URI is only specified in this document, have been added. This document obsoletes [RFC4051]. All of the URIs appear in the indexes in Section 4. Only the URIs that were added by [RFC4051] or this document have a subsection in Section 2 or 3, with the exception of Minimal Canonicalization (Section 2.4), for example, use of

SHA-256 is defined in [XMLENC11] and hence there is no subsection on that algorithm here, but its URI is included in the indexes in Section 4.

Specification in this document of the URI representing an algorithm does not imply endorsement of the algorithm for any particular purpose. A protocol specification, which this is not, generally gives algorithm and implementation requirements for the protocol. Security considerations for algorithms are constantly evolving, as documented elsewhere. This specification simply provides some URIs and relevant formatting for when those URIs are used.

Note that progressing XML Digital Signature [RFC3275] along the Standards Track required removal of any algorithms from the original version [RFC3075] for which there was not demonstrated interoperability. This required removal of the Minimal Canonicalization algorithm, in which there appears to be continued interest. The URI for Minimal Canonicalization was included in [RFC4051] and is included here.

1.1. Terminology

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

This document is not intended to change the algorithm implementation requirements of any IETF or W3C document. Use of [RFC2119] terminology is intended to be only such as is already stated or implied by other authoritative documents.

1.2. Acronyms

The following acronyms are used in this document:

HMAC - Keyed-Hashing MAC [RFC2104]

IETF - Internet Engineering Task Force <www.ietf.org>

MAC - Message Authentication Code

MD - Message Digest

NIST - United States National Institute of Standards and Technology <www.nist.gov>

RC - Rivest Cipher

RSA - Rivest, Shamir, and Adleman

SHA - Secure Hash Algorithm

URI - Uniform Resource Identifier [RFC3986]

W3C - World Wide Web Consortium <www.w3.org>

XML - eXtensible Markup Language

2. Algorithms

The URI [RFC3986] that was dropped from the XML Digital Signature standard due to the transition from Proposed Standard to Draft Standard [RFC3275] is included in Section 2.4 below with its original

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

prefix so as to avoid changing the XMLDSIG standard’s namespace.

Additional algorithms in [RFC4051] were given URIs that start with

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

while further algorithms added in this document are given URIs that start with

```
http://www.w3.org/2007/05/xmldsig-more#
```

In addition, for ease of reference, this document includes in the indexes in Section 4 many cryptographic algorithm URIs from several XML security documents using the namespaces with which they are defined in those documents. For example, 2000/09/xmldsig# for some URIs specified in [RFC3275] and 2001/04/xmlenc# for some URIs specified in [XMLENC10].

See also [XMLSECXREF].

2.1. DigestMethod (Hash) 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 [RFC2045] encoding of this bit string viewed as a 16-octet stream. See [RFC6151] for MD5 security considerations.

2.1.2. SHA-224

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

The SHA-224 algorithm [FIPS180-4] [RFC6234] takes no explicit parameters. An example of a SHA-224 DigestAlgorithm element is:

```
<DigestAlgorithm
   Algorithm="http://www.w3.org/2001/04/xmldsig-more#sha224" />
```

A SHA-224 digest is a 224-bit string. The content of the DigestValue element SHALL be the base64 [RFC2045] encoding of this string viewed as a 28-octet stream.

2.1.3. SHA-384

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

The SHA-384 algorithm [FIPS180-4] takes no explicit parameters. An example of a SHA-384 DigestAlgorithm element is:

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

A SHA-384 digest is a 384-bit string. The content of the DigestValue element SHALL be the base64 [RFC2045] encoding of this string viewed as a 48-octet stream.

2.1.4. Whirlpool

Identifier:
   http://www.w3.org/2007/05/xmldsig-more#whirlpool

The Whirlpool algorithm [10118-3] takes no explicit parameters. A Whirlpool digest is a 512-bit string. The content of the DigestValue element SHALL be the base64 [RFC2045] encoding of this string viewed as a 64-octet stream.

2.1.5. New SHA Functions

Identifiers:
http://www.w3.org/2007/05/xmldsig-more#sha3-224
http://www.w3.org/2007/05/xmldsig-more#sha3-256
http://www.w3.org/2007/05/xmldsig-more#sha3-384
http://www.w3.org/2007/05/xmldsig-more#sha3-512

NIST has recently completed a hash function competition for an alternative to the SHA family. The Keccak-f[1600] algorithm was selected [Keccak] [SHA-3]. This hash function is commonly referred to as "SHA-3", and this section is a space holder and reservation of URIs for future information on Keccak use in XML security.

A SHA-3 224, 256, 384, and 512 digest is a 224-, 256-, 384-, and 512-bit string, respectively. The content of the DigestValue element SHALL be the base64 [RFC2045] encoding of this string viewed as a 28-, 32-, 48-, and 64-octet stream, respectively.

2.2. SignatureMethod MAC Algorithms

This section covers SignatureMethod MAC (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/xmldsig-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 [RFC2045] encoded in the same straightforward fashion as the output of the digest algorithms. 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 above are copied from [RFC3275].

See [RFC6151] for HMAC-MD5 security considerations.

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 [FIPS180-4] [RFC6234] can also be used in HMAC as described in Section 2.2.1 above for HMAC-MD5.

2.2.3. HMAC-RIPEMD160

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

RIPEMD-160 [10118-3] can also be used in HMAC as described in Section 2.2.1 above for HMAC-MD5.

2.3. SignatureMethod Public-Key Signature Algorithms

These algorithms are distinguished from those in Section 2.2 above in that they use public-key methods. That is to say, 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

This 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 [RFC2045] encoding of the octet string computed as per [RFC3447], Section 8.2.1, signature generation for the RSASSA-PKCS1-v1_5 signature scheme. As specified in the EMSA-PKCS1-V1_5-ENCODE function in [RFC3447], Section 9.2, 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 is 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 05 00 04 10
```

This prefix is included to make it easier to use 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.

See [RFC6151] for MD5 security considerations.

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

```
<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

```
<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 it does 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

```
<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*, ECDSA-RIPEMD160, ECDSA-Whirlpool

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
http://www.w3.org/2007/05/xmldsig-more#ecdsa-ripemd160
http://www.w3.org/2007/05/xmldsig-more#ecdsa-whirlpool

The Elliptic Curve Digital Signature Algorithm (ECDSA) [FIPS180-4] is the elliptic curve analogue of the Digital Signature Algorithm (DSA) signature method, i.e., the Digital Signature Standard (DSS). It takes no explicit parameters. For detailed specifications of how to use it with SHA hash functions and XML Digital Signature, please see [X9.62] and [RFC4050]. The #ecdsa-ripemd160 and #ecdsa-whirlpool fragments in the new namespace identifies a signature method processed in the same way as specified by the #ecdsa-sha1 fragment of this namespace, with the exception that RIPEMD160 or Whirlpool is used instead of SHA-1.

The output of the ECDSA algorithm consists of a pair of integers usually referred by the pair (r, s). The signature value consists of the base64 encoding of the concatenation of two octet streams that respectively result from the octet-encoding of the values r and s in that order. Conversion from integer to octet stream must be done according to the I2OSP operation defined in the [RFC3447] specification with the l parameter equal to the size of the base point order of the curve in bytes (e.g., 32 for the P-256 curve and 66 for the P-521 curve [FIPS186-3]).

For an introduction to elliptic curve cryptographic algorithms, see [RFC6090] and note the errata (Errata ID 2773-2777).

2.3.7. ESIGN-SHA*

Identifiers:

```
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 [IEEEP1363a] 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"
/>
```

2.3.8. RSA-Whirlpool

Identifier:

```
http://www.w3.org/2007/05/xmldsig-more#rsa-whirlpool
```

As in the definition of the RSA-SHA1 algorithm in [XMLDSIG11], the designator "RSA" means the RSASSA-PKCS1-v1_5 algorithm as defined in [RFC3447]. When identified through the #rsa-whirlpool fragment identifier, Whirlpool is used as the hash algorithm instead. Use of the ASN.1 BER Whirlpool algorithm designator is implied. That designator is

```
hex 30 4e 30 0a 06 06 28 cf 06 03 00 37 05 00 04 40
```

as an explicit octet sequence. This corresponds to OID 1.0.10118.3.0.55 defined in [10118-3].

An example of use is

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

2.3.9. RSASSA-PSS with Parameters

Identifiers:
http://www.w3.org/2007/05/xmldsig-more#rsa-pss
http://www.w3.org/2007/05/xmldsig-more#MGF1

These identifiers imply the PKCS#1 EMSA-PSS encoding algorithm [RFC3447]. The RSASSA-PSS algorithm takes the digest method (hash function), a mask generation function, the salt length in bytes (SaltLength), and the trailer field as explicit parameters.

Algorithm identifiers for hash functions specified in XML encryption [XMLENC11] [XMLDSIG11] and in Section 2.1 are considered to be valid algorithm identifiers for hash functions. According to [RFC3447], the default value for the digest function is SHA-1, but due to the discovered weakness of SHA-1 [RFC6194], it is recommended that SHA-256 or a stronger hash function be used. Notwithstanding [RFC3447], SHA-256 is the default to be used with these SignatureMethod identifiers if no hash function has been specified.

The default salt length for these SignatureMethod identifiers if the SaltLength is not specified SHALL be the number of octets in the hash value of the digest method, as recommended in [RFC4055]. In a parameterized RSASSA-PSS signature the ds:DigestMethod and the SaltLength parameters usually appear. If they do not, the defaults make this equivalent to http://www.w3.org/2007/05/xmldsig-more#sha256-rsa-MGF1 (see Section 2.3.10). The TrailerField defaults to 1 (0xBC) when omitted.

Schema Definition (target namespace
http://www.w3.org/2007/05/xmldsig-more#):

```
<xs:element name="RSAPSSParams" type="pss:RSAPSSParamsType">
    <xs:annotation>
        <xs:documentation>
Top level element that can be used in xs:any namespace="#other"
wildcard of ds:SignatureMethod content.
        </xs:documentation>
    </xs:annotation>
</xs:element>
<xs:complexType name="RSAPSSParamsType">
    <xs:sequence>
        <xs:element ref="ds:DigestMethod" minOccurs="0"/>
        <xs:element name="MaskGenerationFunction"
           type="pss:MaskGenerationFunctionType" minOccurs="0"/>
        <xs:element name="SaltLength" type="xs:int"
           minOccurs="0"/>
        <xs:element name="TrailerField" type="xs:int"
           minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="MaskGenerationFunctionType">
    <xs:sequence>
        <xs:element ref="ds:DigestMethod" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="Algorithm" type="xs:anyURI"
       default="http://www.w3.org/2007/05/xmldsig-more#MGF1"/>
</xs:complexType>
```

2.3.10. RSASSA-PSS without Parameters

[RFC3447] currently specifies only one mask generation function MGF1 based on a hash function. Although [RFC3447] allows for parameterization, the default is to use the same hash function as the digest method function. Only this default approach is supported by this section; therefore, the definition of a mask generation function type is not needed yet. The same applies to the trailer field. There is only one value (0xBC) specified in [RFC3447]. Hence, this default parameter must be used for signature generation. The default salt length is the length of the hash function.

Identifiers:
http://www.w3.org/2007/05/xmldsig-more#sha3-224-rsa-MGF1
http://www.w3.org/2007/05/xmldsig-more#sha3-256-rsa-MGF1
http://www.w3.org/2007/05/xmldsig-more#sha3-384-rsa-MGF1
http://www.w3.org/2007/05/xmldsig-more#sha3-512-rsa-MGF1

```
http://www.w3.org/2007/05/xmldsig-more#md2-rsa-MGF1
http://www.w3.org/2007/05/xmldsig-more#md5-rsa-MGF1
http://www.w3.org/2007/05/xmldsig-more#sha1-rsa-MGF1
http://www.w3.org/2007/05/xmldsig-more#sha224-rsa-MGF1
http://www.w3.org/2007/05/xmldsig-more#sha256-rsa-MGF1
http://www.w3.org/2007/05/xmldsig-more#sha384-rsa-MGF1
http://www.w3.org/2007/05/xmldsig-more#sha512-rsa-MGF1
http://www.w3.org/2007/05/xmldsig-more#ripemd128-rsa-MGF1
http://www.w3.org/2007/05/xmldsig-more#ripemd160-rsa-MGF1
http://www.w3.org/2007/05/xmldsig-more#whirlpool-rsa-MGF1
```

An example of use is

```
<SignatureMethod
  Algorithm=
  "http://www.w3.org/2007/05/xmldsig-more#SHA3-256-rsa-MGF1"
/>
```

2.3.11. RSA-SHA224

Identifier:
http://www.w3.org/2007/05/xmldsig-more#rsa-sha224

This implies the PKCS#1 v1.5 padding algorithm [RFC3447] as described in Section 2.3.1, but with the ASN.1 BER SHA-224 algorithm designator prefix. An example of use is

```
<SignatureMethod
   Algorithm="http://www.w3.org/2007/05/xmldsig-more#rsa-sha224" />
```

Because it takes about the same effort to calculate a SHA-224 message digest as it does a SHA-256 message digest, it is suggested that RSA-SHA256 be used in preference to RSA-SHA224 where possible.

2.4. Minimal Canonicalization

Thus far, two independent interoperable implementations of Minimal Canonicalization have not been announced. Therefore, when XML Digital Signature was advanced along the Standards Track from [RFC3075] to [RFC3275], Minimal Canonicalization was dropped. However, there is still interest. For its definition, see Section 6.5.1 of [RFC3075].

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:

```
<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:

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

DTD:

```
<!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 [RC4]. An example EncryptionMethod element using ARCFOUR is

```
<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 [XMLENC11].

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 a block cipher with the same interface as the AES [Camellia] [RFC3713]; it has a 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:

```
<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

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 whatever the size of the Camellia key used in wrapping, called the "key encrypting key" or "KEK". If Camellia is supported, it is particularly suggested that wrapping 128-bit keys with a 128-bit KEK and wrapping 256-bit keys with a 256-bit KEK be supported.

An example of use is:

```
<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 [18033-2], is a key encapsulation mechanism using elliptic curve encryption.

An example of use is:

```
<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>
</EncryptionMethod>
```

See [18033-2] for information on the parameters above.

2.6.5. SEED Block Encryption

Identifier:
http://www.w3.org/2007/05/xmldsig-more#seed128-cbc

SEED [RFC4269] is a 128-bit block size with 128-bit key sizes. In XML Encryption, SEED can be 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 SEED EncryptionMethod is as follows:

```
<EncryptionMethod
   Algorithm="http://www.w3.org/2007/05/xmldsig-more#seed128-cbc" />
```

2.6.6. SEED Key Wrap

Identifier:
http://www.w3.org/2007/05/xmldsig-more#kw-seed128

Key wrapping with SEED is identical to Section 2.2.1 of [RFC3394] with "AES" replaced by "SEED". The algorithm is specified in [RFC4010]. The implementation of SEED is optional. The default initial value is 0xA6A6A6A6A6A6A6A6.

An example of use is:

```
<EncryptionMethod
   Algorithm=
   "http://www.w3.org/2007/05/xmldsig-more#kw-seed128"
/>
```

3. KeyInfo

In Section 3.1 below 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 [RFC2045] encoded. Any signer information present is ignored. The following is an example [RFC3092], eliding the base64 data:

```
<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 parsable.

Identifiers:

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

4. Indexes

The following subsections provide an index by URI and by fragment identifier (the portion of the URI after "#") of the algorithm and KeyInfo URIs defined in this document and in the standards (plus the one KeyInfo child element name defined in this document). The "Sec/Doc" column has the section of this document or, if not specified in this document, the document where the item is specified. See also [XMLSECXREF].

4.1. Fragment Index

The initial "http://www.w3.org/" part of the URI is not included below. The first six entries have a null fragment identifier or no fragment identifier.

Fragment	URI	Sec/Doc
	2002/06/xmldsig-filter2	[XPATH]
	2006/12/xmlc12n11#	[CANON11]
	TR/1999/REC-xslt-19991116	[XSLT]
	TR/1999/REC-xpath-19991116	[XPATH]
	TR/2001/06/xml-exc-c14n#	[XCANON]
	TR/2001/REC-xml-c14n-20010315	[CANON10]
	TR/2001/REC-xmlschema-1-20010502	[Schema]
aes128-cbc	2001/04/xmlenc#aes128-cbc	[XMLENC11]
aes128-gcm	2009/xmlenc11#aes128-gcm	[XMLENC11]
aes192-cbc	2001/04/xmlenc#aes192-cbc	[XMLENC11]
aes192-gcm	2009/xmlenc11#aes192-gcm	[XMLENC11]
aes256-cbc	2001/04/xmlenc#aes256-cbc	[XMLENC11]
aes256-gcm	2009/xmlenc11#aes256-gcm	[XMLENC11]
arcfour	2001/04/xmldsig-more#arcfour	2.6.1
base64	2000/09/xmldsig#base64	[RFC3275]
camellia128-cbc	2001/04/xmldsig-more#camellia128-cbc	2.6.2
camellia192-cbc	2001/04/xmldsig-more#camellia192-cbc	2.6.2
camellia256-cbc	2001/04/xmldsig-more#camellia256-cbc	2.6.2
ConcatKDF	2009/xmlenc11#ConcatKDF	[XMLENC11]
decrypt#XML	2002/07/decrypt#XML	[DECRYPT]
decrypt#Binary	2002/07/decrypt#Binary	[DECRYPT]
DEREncodedKeyValue	2009/xmldsig11#DEREncodedKeyValue	[XMLDSIG11]
dh	2001/04/xmlenc#dh	[XMLENC11]
dh-es	2009/xmlenc11#dh-es	[XMLENC11]
dsa-sha1	2000/09/xmldsig#dsa-sha1	[RFC3275]
dsa-sha256	2009/xmldsig11#dsa-sha256	[XMLDSIG11]
DSAKeyValue	2000/09/xmldsig#DSAKeyValue	[XMLDSIG11]
ECDH-ES	2009/xmlenc11#ECDH-ES	[XMLENC11]
ecdsa-ripemd160	2007/05/xmldsig-more#ecdsa-ripemd160	2.3.6
ecdsa-sha1	2001/04/xmldsig-more#ecdsa-sha1	2.3.6
ecdsa-sha224	2001/04/xmldsig-more#ecdsa-sha224	2.3.6
ecdsa-sha256	2001/04/xmldsig-more#ecdsa-sha256	2.3.6
ecdsa-sha384	2001/04/xmldsig-more#ecdsa-sha384	2.3.6
ecdsa-sha512	2001/04/xmldsig-more#ecdsa-sha512	2.3.6

ecdsa-whirlpool	2007/05/xmldsig-more#ecdsa-whirlpool	2.3.5
ecies-kem	2010/xmlsec-ghc#ecies-kem	[GENERIC]
ECKeyValue	2009/xmldsig11#ECKeyValue	[XMLDSIG11]
enveloped-signature	2000/09/xmldsig#enveloped-signature	[RFC3275]
esign-sha1	2001/04/xmldsig-more#esign-sha1	2.3.7
esign-sha224	2001/04/xmldsig-more#esign-sha224	2.3.7
esign-sha256	2001/04/xmldsig-more#esign-sha256	2.3.7
esign-sha384	2001/04/xmldsig-more#esign-sha384	2.3.7
esign-sha512	2001/04/xmldsig-more#esign-sha512	2.3.7
generic-hybrid	2010/xmlsec-ghc#generic-hybrid	[GENERIC]
hmac-md5	2001/04/xmldsig-more#hmac-md5	2.2.1
hmac-ripemd160	2001/04/xmldsig-more#hmac-ripemd160	2.2.3
hmac-sha1	2000/09/xmldsig#hmac-sha1	[RFC3275]
hmac-sha224	2001/04/xmldsig-more#hmac-sha224	2.2.2
hmac-sha256	2001/04/xmldsig-more#hmac-sha256	2.2.2
hmac-sha384	2001/04/xmldsig-more#hmac-sha384	2.2.2
hmac-sha512	2001/04/xmldsig-more#hmac-sha512	2.2.2
KeyName	2001/04/xmldsig-more#KeyName	3.2
KeyValue	2001/04/xmldsig-more#KeyValue	3.2
kw-aes128	2001/04/xmlenc#kw-aes128	[XMLENC11]
kw-aes128-pad	2009/xmlenc11#kw-aes-128-pad	[XMLENC11]
kw-aes192	2001/04/xmlenc#kw-aes192	[XMLENC11]
kw-aes192-pad	2009/xmlenc11#kw-aes-192-pad	[XMLENC11]
kw-aes256	2001/04/xmlenc#kw-aes256	[XMLENC11]
kw-aes256-pad	2009/xmlenc11#kw-aes-256-pad	[XMLENC11]
kw-camellia128	2001/04/xmldsig-more#kw-camellia128	2.6.3
kw-camellia192	2001/04/xmldsig-more#kw-camellia192	2.6.3
kw-camellia256	2001/04/xmldsig-more#kw-camellia256	2.6.3
kw-seed128	2007/05/xmldsig-more#kw-seed128	2.6.6
md2-rsa-MGF1	2007/05/xmldsig-more#md2-rsa-MGF1	2.3.10
md5	2001/04/xmldsig-more#md5	2.1.1
md5-rsa-MGF1	2007/05/xmldsig-more#md5-rsa-MGF1	2.3.10
MGF1	2007/05/xmldsig-more#MGF1	2.3.9
mgf1sha1	2009/xmlenc11#mgf1sha1	[XMLENC11]
mgf1sha224	2009/xmlenc11#mgf1sha224	[XMLENC11]
mgf1sha256	2009/xmlenc11#mgf1sha256	[XMLENC11]
mgf1sha384	2009/xmlenc11#mgf1sha384	[XMLENC11]
mgf1sha512	2009/xmlenc11#mgf1sha512	[XMLENC11]
MgmtData	2000/09/xmldsig#MgmtData	[XMLDSIG11]
minimal	2000/09/xmldsig#minimal	2.4
pbkdf2	2009/xmlenc11#pbkdf2	[XMLENC11]
PGPData	2000/09/xmldsig#PGPData	[XMLDSIG11]
PKCS7signedData	2001/04/xmldsig-more#PKCS7signedData	3.1

PKCS7signedData	2001/04/xmldsig-more#PKCS7signedData	3.2
psec-kem	2001/04/xmldsig-more#psec-kem	2.6.4

rawPGPKeyPacket	2001/04/xmldsig-more#rawPGPKeyPacket	3.2
rawPKCS7signedData	2001/04/xmldsig-more#rawPKCS7signedData	3.2
rawSPKISexp	2001/04/xmldsig-more#rawSPKISexp	3.2
rawX509Certificate	2000/09/xmldsig#rawX509Certificate	[RFC3275]
rawX509CRL	2001/04/xmldsig-more#rawX509CRL	3.2
RetrievalMethod	2001/04/xmldsig-more#RetrievalMethod	3.2
ripemd128-rsa-MGF1	2007/05/xmldsig-more#ripemd128-rsa-MGF1	2.3.10
ripemd160	2001/04/xmlenc#ripemd160	[XMLENC11]
ripemd160-rsa-MGF1	2007/05/xmldsig-more#ripemd160-rsa-MGF1	2.3.10
rsa-1_5	2001/04/xmlenc#rsa-1_5	[XMLENC11]
rsa-md5	2001/04/xmldsig-more#rsa-md5	2.3.1
rsa-oaep	2009/xmlenc11#rsa-oaep	[XMLENC11]
rsa-oaep-mgf1p	2001/04/xmlenc#rsa-oaep-mgf1p	[XMLENC11]
rsa-pss	2007/05/xmldsig-more#rsa-pss	2.3.9
rsa-ripemd160	2001/04/xmldsig-more#rsa-ripemd160	2.3.5
rsa-sha1	2000/09/xmldsig#rsa-sha1	[RFC3275]
rsa-sha224	2007/05/xmldsig-more#rsa-sha224	2.3.11
rsa-sha256	2001/04/xmldsig-more#rsa-sha256	2.3.2
rsa-sha384	2001/04/xmldsig-more#rsa-sha384	2.3.3
rsa-sha512	2001/04/xmldsig-more#rsa-sha512	2.3.4
rsa-whirlpool	2007/05/xmldsig-more#rsa-whirlpool	2.3.5
rsaes-kem	2010/xmlsec-ghc#rsaes-kem	[GENERIC]
RSAKeyValue	2000/09/xmldsig#RSAKeyValue	[XMLDSIG11]

seed128-cbc	2007/05/xmldsig-more#seed128-cbc	2.6.5
sha1	2000/09/xmldsig#sha1	[RFC3275]
sha1-rsa-MGF1	2007/05/xmldsig-more#sha1-rsa-MGF1	2.3.10
sha224	2001/04/xmldsig-more#sha224	2.1.2
sha224-rsa-MGF1	2007/05/xmldsig-more#sha224-rsa-MGF1	2.3.10
sha256	2001/04/xmlenc#sha256	[XMLENC11]
sha256-rsa-MGF1	2007/05/xmldsig-more#sha256-rsa-MGF1	2.3.10
sha3-224	2007/05/xmldsig-more#sha3-224	2.1.5
sha3-224-rsa-MGF1	2007/05/xmldsig-more#sha3-224-rsa-MGF1	2.3.10
sha3-256	2007/05/xmldsig-more#sha3-256	2.1.5
sha3-256-rsa-MGF1	2007/05/xmldsig-more#sha3-256-rsa-MGF1	2.3.10
sha3-384	2007/05/xmldsig-more#sha3-384	2.1.5
sha3-384-rsa-MGF1	2007/05/xmldsig-more#sha3-384-rsa-MGF1	2.3.10
sha3-512	2007/05/xmldsig-more#sha3-512	2.1.5
sha3-512-rsa-MGF1	2007/05/xmldsig-more#sha3-512-rsa-MGF1	2.3.10
sha384	2001/04/xmldsig-more#sha384	2.1.3
sha384-rsa-MGF1	2007/05/xmldsig-more#sha384-rsa-MGF1	2.3.10
sha512	2001/04/xmlenc#sha512	[XMLENC11]
sha512-rsa-MGF1	2007/05/xmldsig-more#sha512-rsa-MGF1	2.3.10
SPKIData	2000/09/xmldsig#SPKIData	[XMLDSIG11]

tripledes-cbc	2001/04/xmlenc#tripledes-cbc	[XMLENC11]
whirlpool	2007/05/xmldsig-more#whirlpool	2.1.4
whirlpool-rsa-MGF1	2007/05/xmldsig-more#whirlpool-rsa-MGF1	2.3.10
WithComments	2006/12/xmlc14n11#WithComments	[CANON11]
WithComments	TR/2001/06/xml-exc-c14n#WithComments	[XCANON]
WithComments	TR/2001/REC-xml-c14n-20010315#WithComments	[CANON10]
X509Data	2000/09/xmldsig#X509Data	[XMLDSIG11]
xptr	2001/04/xmldsig-more#xptr	2.5.1

The initial "http://www.w3.org/" part of the URI is not included above.

4.2. URI Index

The initial "http://www.w3.org/" part of the URI is not included below.

URI	Sec/Doc	Type
2000/09/xmldsig#base64	[RFC3275]	Transform
2000/09/xmldsig#DSAKeyValue	[RFC3275]	Retrieval type
2000/09/xmldsig#dsa-sha1	[RFC3275]	SignatureMethod
2000/09/xmldsig#enveloped-signature	[RFC3275]	Transform
2000/09/xmldsig#hmac-sha1	[RFC3275]	SignatureMethod
2000/09/xmldsig#MgmtData	[RFC3275]	Retrieval type
2000/09/xmldsig#minimal	2.4	Canonicalization
2000/09/xmldsig#PGPData	[RFC3275]	Retrieval type
2000/09/xmldsig#rawX509Certificate	[RFC3275]	Retrieval type
2000/09/xmldsig#rsa-sha1	[RFC3275]	SignatureMethod
2000/09/xmldsig#RSAKeyValue	[RFC3275]	Retrieval type
2000/09/xmldsig#sha1	[RFC3275]	DigestAlgorithm
2000/09/xmldsig#SPKIData	[RFC3275]	Retrieval type
2000/09/xmldsig#X509Data	[RFC3275]	Retrieval type
2001/04/xmldsig-more#arcfour	2.6.1	EncryptionMethod
2001/04/xmldsig-more#camellia128-cbc	2.6.2	EncryptionMethod
2001/04/xmldsig-more#camellia192-cbc	2.6.2	EncryptionMethod
2001/04/xmldsig-more#camellia256-cbc	2.6.2	EncryptionMethod
2001/04/xmldsig-more#ecdsa-sha1	2.3.6	SignatureMethod
2001/04/xmldsig-more#ecdsa-sha224	2.3.6	SignatureMethod
2001/04/xmldsig-more#ecdsa-sha256	2.3.6	SignatureMethod
2001/04/xmldsig-more#ecdsa-sha384	2.3.6	SignatureMethod
2001/04/xmldsig-more#ecdsa-sha512	2.3.6	SignatureMethod
2001/04/xmldsig-more#esign-sha1	2.3.7	SignatureMethod

2001/04/xmldsig-more#esign-sha224	2.3.7	SignatureMethod
2001/04/xmldsig-more#esign-sha256	2.3.7	SignatureMethod
2001/04/xmldsig-more#esign-sha384	2.3.7	SignatureMethod
2001/04/xmldsig-more#esign-sha512	2.3.7	SignatureMethod
2001/04/xmldsig-more#hmac-md5	2.2.1	SignatureMethod
2001/04/xmldsig-more#hmac-ripemd160	2.2.3	SignatureMethod
2001/04/xmldsig-more#hmac-sha224	2.2.2	SignatureMethod
2001/04/xmldsig-more#hmac-sha256	2.2.2	SignatureMethod
2001/04/xmldsig-more#hmac-sha384	2.2.2	SignatureMethod
2001/04/xmldsig-more#hmac-sha512	2.2.2	SignatureMethod
2001/04/xmldsig-more#KeyName	3.2	Retrieval type
2001/04/xmldsig-more#KeyValue	3.2	Retrieval type
2001/04/xmldsig-more#kw-camellia128	2.6.3	EncryptionMethod
2001/04/xmldsig-more#kw-camellia192	2.6.3	EncryptionMethod
2001/04/xmldsig-more#kw-camellia256	2.6.3	EncryptionMethod
2001/04/xmldsig-more#md5	2.1.1	DigestAlgorithm
2001/04/xmldsig-more#PKCS7signedData	3.2	Retrieval type
2001/04/xmldsig-more#psec-kem	2.6.4	EncryptionMethod
2001/04/xmldsig-more#rawPGPKeyPacket	3.2	Retrieval type
2001/04/xmldsig-more#rawPKCS7signedData	3.2	Retrieval type
2001/04/xmldsig-more#rawSPKISexp	3.2	Retrieval type
2001/04/xmldsig-more#rawX509CRL	3.2	Retrieval type
2001/04/xmldsig-more#RetrievalMethod	3.2	Retrieval type
2001/04/xmldsig-more#rsa-md5	2.3.1	SignatureMethod
2001/04/xmldsig-more#rsa-sha256	2.3.2	SignatureMethod
2001/04/xmldsig-more#rsa-sha384	2.3.3	SignatureMethod
2001/04/xmldsig-more#rsa-sha512	2.3.4	SignatureMethod
2001/04/xmldsig-more#rsa-ripemd160	2.3.5	SignatureMethod
2001/04/xmldsig-more#sha224	2.1.2	DigestAlgorithm
2001/04/xmldsig-more#sha384	2.1.3	DigestAlgorithm
2001/04/xmldsig-more#xptr	2.5.1	Transform
2001/04/xmldsig-more#PKCS7signedData	3.1	KeyInfo child
2001/04/xmlenc#aes128-cbc	[XMLENC11]	EncryptionMethod
2001/04/xmlenc#aes192-cbc	[XMLENC11]	EncryptionMethod
2001/04/xmlenc#aes256-cbc	[XMLENC11]	EncryptionMethod
2001/04/xmlenc#dh	[XMLENC11]	AgreementMethod
2001/04/xmlenc#kw-aes128	[XMLENC11]	EncryptionMethod
2001/04/xmlenc#kw-aes192	[XMLENC11]	EncryptionMethod
2001/04/xmlenc#kw-aes256	[XMLENC11]	EncryptionMethod
2001/04/xmlenc#ripemd160	[XMLENC11]	DigestAlgorithm
2001/04/xmlenc#rsa-1_5	[XMLENC11]	EncryptionMethod
2001/04/xmlenc#rsa-oaep-mgf1p	[XMLENC11]	EncryptionMethod
2001/04/xmlenc#sha256	[XMLENC11]	DigestAlgorithm
2001/04/xmlenc#sha512	[XMLENC11]	DigestAlgorithm
2001/04/xmlenc#tripledes-cbc	[XMLENC11]	EncryptionMethod
2002/06/xmldsig-filter2	[XPATH]	Transform

2002/07/decrypt#XML	[DECRYPT]	Transform
2002/07/decrypt#Binary	[DECRYPT]	Transform

2006/12/xmlc12n11#	[CANON11]	Canonicalization
2006/12/xmlc14n11#WithComments	[CANON11]	Canonicalization

2007/05/xmldsig-more#ecdsa-ripemd160	2.3.6	SignatureMethod
2007/05/xmldsig-more#ecdsa-whirlpool	2.3.5	SignatureMethod
2007/05/xmldsig-more#kw-seed128	2.6.6	EncryptionMethod
2007/05/xmldsig-more#md2-rsa-MGF1	2.3.10	SignatureMethod
2007/05/xmldsig-more#md5-rsa-MGF1	2.3.10	SignatureMethod
2007/05/xmldsig-more#MGF1	2.3.9	SignatureMethod
2007/05/xmldsig-more#ripemd128-rsa-MGF1	2.3.10	SignatureMethod
2007/05/xmldsig-more#ripemd160-rsa-MGF1	2.3.10	SignatureMethod
2007/05/xmldsig-more#rsa-pss	2.3.9	SignatureMethod
2007/05/xmldsig-more#rsa-sha224	2.3.11	SignatureMethod
2007/05/xmldsig-more#rsa-whirlpool	2.3.5	SignatureMethod
2007/05/xmldsig-more#seed128-cbc	2.6.5	EncryptionMethod
2007/05/xmldsig-more#sha1-rsa-MGF1	2.3.10	SignatureMethod
2007/05/xmldsig-more#sha224-rsa-MGF1	2.3.10	SignatureMethod
2007/05/xmldsig-more#sha256-rsa-MGF1	2.3.10	SignatureMethod
2007/05/xmldsig-more#sha3-224	2.1.5	DigestAlgorithm
2007/05/xmldsig-more#sha3-224-rsa-MGF1	2.3.10	SignatureMethod
2007/05/xmldsig-more#sha3-256	2.1.5	DigestAlgorithm
2007/05/xmldsig-more#sha3-256-rsa-MGF1	2.3.10	SignatureMethod
2007/05/xmldsig-more#sha3-384	2.1.5	DigestAlgorithm
2007/05/xmldsig-more#sha3-384-rsa-MGF1	2.3.10	SignatureMethod
2007/05/xmldsig-more#sha3-512	2.1.5	DigestAlgorithm
2007/05/xmldsig-more#sha3-512-rsa-MGF1	2.3.10	SignatureMethod
2007/05/xmldsig-more#sha384-rsa-MGF1	2.3.10	SignatureMethod
2007/05/xmldsig-more#sha512-rsa-MGF1	2.3.10	SignatureMethod
2007/05/xmldsig-more#whirlpool	2.1.4	DigestAlgorithm
2007/05/xmldsig-more#whirlpool-rsa-MGF1	2.3.10	SignatureMethod
2009/xmlenc11#kw-aes-128-pad	[XMLENC11]	EncryptionMethod
2009/xmlenc11#kw-aes-192-pad	[XMLENC11]	EncryptionMethod
2009/xmlenc11#kw-aes-256-pad	[XMLENC11]	EncryptionMethod

2009/xmldsig11#dsa-sha256	[XMLDSIG11]	SignatureMethod
2009/xmldsig11#ECKeyValue	[XMLDSIG11]	Retrieval type
2009/xmldsig11#DEREncodedKeyValue	[XMLDSIG11]	Retrieval type

2009/xmlenc11#aes128-gcm	[XMLENC11]	EncryptionMethod
2009/xmlenc11#aes192-gcm	[XMLENC11]	EncryptionMethod
2009/xmlenc11#aes256-gcm	[XMLENC11]	EncryptionMethod
2009/xmlenc11#ConcatKDF	[XMLENC11]	EncryptionMethod
2009/xmlenc11#mgf1sha1	[XMLENC11]	SignatureMethod
2009/xmlenc11#mgf1sha224	[XMLENC11]	SignatureMethod
2009/xmlenc11#mgf1sha256	[XMLENC11]	SignatureMethod

2009/xmlenc11#mgf1sha384	[XMLENC11]	SignatureMethod
2009/xmlenc11#mgf1sha512	[XMLENC11]	SignatureMethod
2009/xmlenc11#pbkdf2	[XMLENC11]	EncryptionMethod
2009/xmlenc11#rsa-oaep	[XMLENC11]	EncryptionMethod
2009/xmlenc11#ECDH-ES	[XMLENC11]	EncryptionMethod
2009/xmlenc11#dh-es	[XMLENC11]	EncryptionMethod

2010/xmlsec-ghc#generic-hybrid	[GENERIC]	Generic Hybrid
2010/xmlsec-ghc#rsaes-kem	[GENERIC]	Generic Hybrid
2010/xmlsec-ghc#ecies-kem	[GENERIC]	Generic Hybrid

TR/1999/REC-xpath-19991116	[XPATH]	Transform
TR/1999/REC-xslt-19991116	[XSLT]	Transform
TR/2001/06/xml-exc-c14n#	[XCANON]	Canonicalization
TR/2001/06/xml-exc-c14n#WithComments	[XCANON]	Canonicalization
TR/2001/REC-xml-c14n-20010315	[CANON10]	Canonicalization
TR/2001/REC-xml-c14n-20010315#WithComments	[CANON10]	Canonicalization
TR/2001/REC-xmlschema-1-20010502	[Schema]	Transform

The initial "http://www.w3.org/" part of the URI is not included above.

5. Allocation Considerations

W3C and IANA allocation considerations are given below.

5.1. W3C Allocation Considerations

As it is easy for people to construct their own unique URIs [RFC3986] and, if appropriate, to obtain a URI from the W3C, it is not intended that any additional "http://www.w3.org/2007/05/xmldsig-more#" URIs be created beyond those enumerated in this RFC. (W3C Namespace stability rules prohibit the creation of new URIs under "http://www.w3.org/2000/09/xmldsig#" and URIs under "http://www.w3.org/2001/04/xmldsig-more#" were frozen with the publication of [RFC4051].)

An "xmldsig-more" URI does not imply any official W3C or IETF status for these algorithms or identifiers nor does it imply that they are only useful in digital signatures. Currently, dereferencing such URIs may or may not produce a temporary placeholder document. Permission to use these URI prefixes has been given by the W3C.

5.2. IANA Considerations

IANA has established a registry entitled "XML Security URIs". The initial contents correspond to Section 4.2 of this document with each section number in the "Sec/Doc" column augmented with a reference to this RFC (for example, "2.6.4" means "[RFC6931], Section 2.6.4").

New entries, including new Types, will be added based on Expert Review [RFC5226]. Criterion for inclusion are (1) documentation sufficient for interoperability of the algorithm or data type and the XML syntax for its representation and use and (2) sufficient importance as normally indicated by inclusion in (2a) an approved W3C Note, Proposed Recommendation, or Recommendation or (2b) an approved IETF Standards Track document. Typically, the registry will reference a W3C or IETF document specifying such XML syntax; that document will either contain a more abstract description of the algorithm or data type or reference another document with a more abstract description.

6. Security Considerations

This RFC is concerned with documenting the URIs that designate algorithms and some data types used in connection with XML security. The security considerations vary widely with the particular algorithms, and the general security considerations for XML security are outside of the scope of this document but appear in [XMLDSIG11], [XMLENC11], [CANON10], [CANON11], and [GENERIC].

[RFC6151] should be consulted before considering the use of MD5 as a DigestMethod or RSA-MD5 as a SignatureMethod.

See [RFC6194] for SHA-1 security considerations and [RFC6151] for MD5 security considerations.

Additional security considerations are given in connection with the description of some algorithms in the body of this document.

Implementers should be aware that cryptographic algorithms become weaker with time. As new cryptoanalysis techniques are developed and computing performance improves, the work factor to break a particular cryptographic algorithm will reduce. Therefore, cryptographic implementations should be modular, allowing new algorithms to be readily inserted. That is, implementers should be prepared for the set of mandatory-to-implement algorithms to change over time.

7. Acknowledgements

The contributions to this document by the following people, listed in alphabetic order, are gratefully acknowledged: Benoit Claise, Adrian Farrel, Stephen Farrell, Ernst Giessmann, Frederick Hirsch, Bjoern Hoehrmann, Russ Housley, Satoru Kanno, Charlie Kaufman, Konrad Lanz, HwanJin Lee, Barry Leiba, Peter Lipp, Subramanian Moonesamy, Thomas Roessler, Hanseong Ryu, Peter Saint-Andre, and Sean Turner.

The following contributors to [RFC4051], on which this document is based, are gratefully acknowledged: Glenn Adams, Merlin Hughs, Gregor Karlinger, Brian LaMachia, Shiho Moriai, Joseph Reagle, Russ Housley, and Joel Halpern.

Appendix A. Changes from RFC 4051

The following changes have been made in RFC 4051 to produce this document.

1. Updated and added numerous RFC, W3C, and Internet-Draft references.

2. Added #ecdsa-ripemd160, #whirlpool, #ecdsa-whirlpool, #rsa-whirlpool, #seed128-cbc, and #kw-seed128.

3. Incorporated RFC 4051 errata [Errata191].

4. Added URI and fragment index sections.

5. For MD5 and SHA-1, added references to [RFC6151] and [RFC6194].

5. Added SHA-3 / Keccak placeholder section including #sha3-224, #sha3-256, #sha3-384, and #sha3-512.

6. Added RSASSA-PSS sections including #sha3-224-MGF1, #sha3-256-MGF1, #sha3-384-MGF1, #sha3-512-MGF1, #md2-rsa-MGF1, #md5-rsa-MGF1, #sha1-rsa-MGF1, #sha224-rsa-MGF1, #sha256-rsa-MGF1, #sha384-rsa-MGF1, #sha512-rsa-MGF1, #ripemd128-rsa-MGF1, #ripemd160-rsa-MGF1, and #whirlpool-rsa-MGF1.

7. Added new URIs from Canonical XML 1.1 and XML Encryption 1.1 including: #aes128-gcm, #aes192-gcm, #aes256-gc, #ConcatKDF, #pbkdf, #rsa-oaep, #ECDH-ES, and #dh-es.

8. Added acronym subsection.

9. Added numerous URIs that are specified in W3C XML Security documents to the Indexes. These do not have sections in the body of this document -- for example, those for dsa-sha256, mgf1sha*, decrypt#XML, and xmldsig-filter2.

10. Requested establishment of an IANA registry.

11. Made various editorial changes.

Normative References

[10118-3]	ISO, "Information technology -- Security techniques -- Hash-functions -- Part 3: Dedicated hash-functions", ISO/IEC 10118-3:2004, 2004.
[18033-2]	ISO, "Information technology -- Security techniques -- Encryption algorithms -- Part 3: Asymmetric ciphers", ISO/IEC 18033-2:2010, 2010.
[Camellia]	Aoki, K., Ichikawa, T., Matsui, M., Moriai, S., Nakajima, J., and T. Tokita, "Camellia: A 128-bit Block Cipher Suitable for Multiple Platforms - Design and Analysis", in Selected Areas in Cryptography, 7th Annual International Workshop, SAC 2000, August 2000, Proceedings, Lecture Notes in Computer Science 2012, pp. 39-56, Springer-Verlag, 2001.
[FIPS180-4]	US National Institute of Science and Technology, "Secure Hash Standard (SHS)", FIPS 180-4, March 2012, <http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf>.
[FIPS186-3]	US National Institute of Science and Technology, "Digital Signature Standard (DSS)", FIPS 186-3, June 2009, <http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf>.
[IEEEP1363a]	IEEE, "Standard Specifications for Public Key Cryptography- Amendment 1: Additional Techniques", IEEE 1363a-2004, 2004.
[RC4]	Schneier, B., "Applied Cryptography: Protocols, Algorithms, and Source Code in C", Second Edition, John Wiley and Sons, New York, NY, 1996.
[RFC1321]	Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.
[RFC2045]	Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.
[RFC2104]	Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2315] Kaliski, B., "PKCS #7: Cryptographic Message Syntax Version 1.5", RFC 2315, March 1998.

[RFC3275] Eastlake 3rd, D., Reagle, J., and D. Solo, "(Extensible Markup Language) XML-Signature Syntax and Processing", RFC 3275, March 2002.

[RFC3394] Schaad, J. and R. Housley, "Advanced Encryption Standard (AES) Key Wrap Algorithm", RFC 3394, September 2002.

[RFC3447] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, February 2003.

[RFC3713] Matsui, M., Nakajima, J., and S. Moriai, "A Description of the Camellia Encryption Algorithm", RFC 3713, April 2004.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4050] Blake-Wilson, S., Karlinger, G., Kobayashi, T., and Y. Wang, "Using the Elliptic Curve Signature Algorithm (ECDSA) for XML Digital Signatures", RFC 4050, April 2005.

[RFC4055] Schaad, J., Kaliski, B., and R. Housley, "Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 4055, June 2005.

[RFC4269] Lee, H., Lee, S., Yoon, J., Cheon, D., and J. Lee, "The SEED Encryption Algorithm", RFC 4269, December 2005.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, May 2011.

[X9.62] American National Standards Institute, Accredited Standards Committee X9, "Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)", ANSI X9.62:2005, 2005.

[XMLENC10] Reagle, J. and D. Eastlake, "XML Encryption Syntax and Processing", W3C Recommendation, 10 December 2002, <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/>.

[XMLENC11] Eastlake, D., Reagle, J., Hirsch, F., and T. Roessler, "XML Encryption Syntax and Processing Version 1.1", W3C Proposed Recommendation, 24 January 2013, <http://www.w3.org/TR/2013/PR-xmlenc-core1-20130124/>.

[XPointer] Grosso, P., Maler, E., Marsh, J., and N. Walsh, "XPointer Framework", W3C Recommendation, 25 March 2003, <http://www.w3.org/TR/2003/REC-xptr-framework-20030325/>.

Informative References

[CANON10] Boyer, J., "Canonical XML Version 1.0", W3C Recommendation, 15 March 2001, <http://www.w3.org/TR/2001/REC-xml-c14n-20010315>.

[CANON11] Boyer, J., and G. Marcy, "Canonical XML Version 1.1", W3C Recommendation, 2 May 2008, <http://www.w3.org/TR/2008/REC-xml-c14n11-20080502/>.

[DECRYPT] Hughes, M., Imamura, T., and H. Maruyama, "Decryption Transform for XML Signature", W3C Recommendation, 10 December 2002, <http://www.w3.org/TR/2002/REC-xmlenc-decrypt-20021210>.

[Errata191] RFC Errata, Errata ID 191, RFC 4051, <http://www.rfc-editor.org>.

[GENERIC] Nystrom, M. and F. Hirsch, "XML Security Generic Hybrid Ciphers", W3C Working Group Note, 24 January 2013, <http://www.w3.org/TR/2013/NOTE-xmlsec-generic-hybrid-20130124/>.

[Keccak] Bertoni, G., Daeman, J., Peeters, M., and G. Van Assche, "The KECCAK sponge function family", January 2013, <http://keccak.noekeon.org>.

[RFC3075] Eastlake 3rd, D., Reagle, J., and D. Solo, "XML-Signature Syntax and Processing", RFC 3075, March 2001.

[RFC3076] Boyer, J., "Canonical XML Version 1.0", RFC 3076, March 2001.

[RFC3092] Eastlake 3rd, D., Manros, C., and E. Raymond, "Etymology of "Foo"", RFC 3092, 1 April 2001.

[RFC3741] Boyer, J., Eastlake 3rd, D., and J. Reagle, "Exclusive XML Canonicalization, Version 1.0", RFC 3741, March 2004.

[RFC4010] Park, J., Lee, S., Kim, J., and J. Lee, "Use of the SEED Encryption Algorithm in Cryptographic Message Syntax (CMS)", RFC 4010, February 2005.

[RFC4051] Eastlake 3rd, D., "Additional XML Security Uniform Resource Identifiers (URIs)", RFC 4051, April 2005.

[RFC6090] McGrew, D., Igoe, K., and M. Salter, "Fundamental Elliptic Curve Cryptography Algorithms", RFC 6090, February 2011.

[RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, March 2011.

[RFC6194] Polk, T., Chen, L., Turner, S., and P. Hoffman, "Security Considerations for the SHA-0 and SHA-1 Message-Digest Algorithms", RFC 6194, March 2011.

[Schema] Thompson, H., Beech, D., Maloney, M., and N. Mendelsohn, "XML Schema Part 1: Structures Second Edition", W3C Recommendation, 28 October 2004, <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/>.

Biron, P. and A. Malhotra, "XML Schema Part 2: Datatypes Second Edition", W3C Recommendation, 28 October 2004, <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/>.

[SHA-3] US National Institute of Science and Technology, "SHA-3 WINNER", February 2013, <http://csrc.nist.gov/groups/ST/hash/sha-3/winner_sha-3.html>.

[W3C] World Wide Web Consortium, <http://www.w3.org>.

[XCANON] Boyer, J., Eastlake, D., and J. Reagle, "Exclusive XML Canonicalization Version 1.0", W3C Recommendation, 18 July 2002, <http://www.w3.org/TR/2002/REC-xml-exc-c14n-20020718/>.

[XMLDSIG10] Eastlake, D., Reagle, J., Solo, D., Hirsch, F., and T. Roessler, "XML Signature Syntax and Processing (Second Edition)", W3C Recommendation, 10 June 2008, <http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/>.

[XMLDSIG11] Eastlake, D., Reagle, J., Solo, D., Hirsch, F., Nystrom, M., Roessler, T., and K. Yiu, "XML Signature Syntax and Processing Version 1.1", W3C Proposed Recommendation, 24 January 2013, <http://www.w3.org/TR/2013/PR-xmldsig-core1-20130124/>.

[XMLDSIG-PROP] Hirsch, F., "XML Signature Properties", W3C Proposed Recommendation, 24 January 2013, <http://www.w3.org/TR/2013/PR-xmldsig-properties-20130124/>.

[XMLSECXREF] Hirsch, F., Roessler, T., and K. Yiu, "XML Security Algorithm Cross-Reference", W3C Working Group Note, 24 January 2013, <http://www.w3.org/TR/2013/NOTE-xmlsec-algorithms-20130124/>.

[XPATH] Boyer, J., Hughes, M., and J. Reagle, "XML-Signature XPath Filter 2.0", W3C Recommendation, 8 November 2002, <http://www.w3.org/TR/2002/REC-xmldsig-filter2-20021108/>.

Berglund, A., Boag, S., Chamberlin, D., Fernandez, M., Kay, M., Robie, J., and J. Simeon, "XML Path Language (XPath) 2.0 (Second Edition)", W3C Recommendation, 14 December 2010, <http://www.w3.org/TR/2010/REC-xpath20-20101214/>.

[XSLT] Saxonica, M., "XSL Transformations (XSLT) Version 2.0", W3C Recommendation, 23 January 2007, <http://www.w3.org/TR/2007/REC-xslt20-20070123/>.

Author's Address

Donald E. Eastlake, 3rd
Huawei Technologies
155 Beaver Street
Milford, MA 01757
USA

Phone: +1-508-333-2270
EMail: d3e3e3@gmail.com