Internet Engineering Task Force (IETF)
Request for Comments: 9231
Obsoletes: 6931
Category: Standards Track
ISSN: 2070-1721

D. Eastlake 3rd
Futurewei Technologies, Inc.

July 2022

Additional XML Security Uniform Resource Identifiers (URIs)

Abstract

This document updates and corrects the IANA "XML Security URIs" registry that lists URIs intended for use with XML digital signatures, encryption, canonicalization, and key management. These URIs identify algorithms and types of information. This document also obsoletes and corrects three errata against RFC 6931.

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 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9231.

Table of Contents

1. Introduction

XML digital signatures, canonicalization, and encryption were standardized by the W3C and by the joint IETF/W3C XMLDSIG working group [W3C] [XMLSEC]. 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

Table 1

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].

In addition, the XML Encryption recommendation has been augmented by [GENERIC], which defines algorithms, XML types, and elements necessary to use generic hybrid ciphers in XML security applications. [GENERIC] also provides for a key encapsulation algorithm and a data encapsulation algorithm, with the combination of the two forming the generic hybrid cipher.

All camel-case element names (names with both interior upper and lower case letters) 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. This document fixes Errata [Err3597], [Err3965], and [Err4004], and obsoletes [RFC6931].

All of the URIs for algorithms and data types herein are listed in the indexes in Section 4. Of these URIs, those that were added by earlier RFCs or by this document have a subsection in Section 2 or 3. A few URIs defined elsewhere also have a subsection in Section 2 or 3, but most such URIs do not. For example, use of SHA-256 as defined in [XMLENC11] has no subsection here but 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 when those URIs are used.

This document is not intended to change the algorithm implementation requirements of any IETF or W3C document. Use of terminology from [RFC2119] and [RFC8174] is intended to be only such as is already stated or implied by other authoritative documents.

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 was continued interest. The URI for Minimal Canonicalization was included in [RFC6931] 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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

"camel-case" refers to terms that are mostly lower case but have internal capital letters.

1.2. Acronyms

The following acronyms are used in this document:

AAD - Additional Authenticated Data

AEAD - Authenticated Encryption with Associated Data

ASN.1 - Abstract Syntax Notation 1

BER - Basic Encoding Rules [ITU-T-X.680]

DSA - Digital Signature Algorithm

DSS - Digital Signature Standard [FIPS186-4]

ECDSA - Elliptic Curve DSA

HMAC - Hashed Message Authentication Code [RFC2104] [RFC5869]

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

MAC - Message Authentication Code

MD - Message Digest

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

OID - Object Identifier [ITU-T-X.660]

PKCS - Public Key Cryptography Standard

RSA - Rivest, Shamir, and Adleman

SHA - Secure Hash Algorithm

URI - Uniform Resource Identifier [RFC3986]

W3C - World Wide Web Consortium <https://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 with its original

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

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

Additional algorithms in RFC 4051 were given URIs that start with

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

Further algorithms added in [RFC6931] were given URIs that start with

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

and algorithms added in this document are given URIs that start with

http://www.w3.org/2021/04/xmldsig-more#

In addition, for ease of reference, this document includes in the indexes in Section 4 many cryptographic algorithm URIs from XML security documents using the namespaces with which they are defined in those documents as follows:

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

for some URIs specified in [RFC3275],

http://www.w3.org/2001/04/xmlenc#

for some URIs specified in [XMLENC10], and

http://www.w3/org/xmlsec-ghc#

for some URIs specified in [GENERIC].

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 [RFC4648] 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 [RFC4648] 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 [RFC4648] 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 [ISO-10118-3] takes no explicit parameters. An example of a Whirlpool DigestAlgorithm element is:

```
<DigestAlgorithm
   Algorithm="http://www.w3.org/2007/05/xmldsig-more#whirlpool" />
```

A Whirlpool digest is a 512-bit string. The content of the DigestValue element SHALL be the base64 [RFC4648] encoding of this string viewed as a 64-octet stream.

2.1.5. SHA-3 Algorithms

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 conducted a hash function competition for an alternative to the SHA family. The Keccak-f[1600] algorithm was selected [KECCAK]. This hash function is commonly referred to as "SHA-3" [FIPS202].

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 [RFC4648] encoding of this string viewed as a 28-, 32-, 48-, and 64-octet stream, respectively. An example of a SHA3-224 DigestAlgorithm element is:

```
<DigestAlgorithm
   Algorithm="http://www.w3.org/2007/05/xmldsig-more#sha3-224" />
```

2.2. SignatureMethod MAC Algorithms

This section covers SignatureMethod Message Authentication Code (MAC) Algorithms.

Note: Some text in this section is duplicated from [RFC3275] for the convenience of the reader. [RFC3275] 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 the output (possibly truncated) of the chosen digest algorithm. This value SHALL be base64 [RFC4648] 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 for HMAC-MD5.

2.2.3. HMAC-RIPEMD160

Identifier:

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

RIPEMD-160 [ISO-10118-3] is a 160-bit hash that is used here in HMAC. The output can be optionally truncated. An example is as follows:

```
<SignatureMethod
   Algorithm="http://www.w3.org/2001/04/xmldsig-more#hmac-ripemd160">
   <HMACOutputLength>144</HMACOutputLength>
</SignatureMethod>
```

2.2.4. Poly1305

Identifier:

```
http://www.w3.org/2021/04/xmldsig-more#poly1305
```

Poly1305 [RFC8439] [POLY1305] is a high-speed message authentication code algorithm. It takes a 32-octet one-time key and a message and produces a 16-octet tag, which is used to authenticate the message. An example of a Poly1305 SignatureMethod element is as follows:

```
<SignatureMethod
   Algorithm="http://www.w3.org/2021/04/xmldsig-more#poly1305"/>
```

2.2.5. SipHash-2-4

Identifier:
http://www.w3.org/2021/04/xmldsig-more#siphash-2-4

SipHash [SipHash1] [SipHash2] computes a 64-bit MAC from a 128-bit secret key and a variable-length message. An example of a SipHash-2-4 SignatureMethod element is as follows:

```
<SignatureMethod
   Algorithm="http://www.w3.org/2021/04/xmldsig-more#siphash-2-4"/>
```

2.2.6. XMSS and XMSSMT

XMSS (eXtended Merkle Signature Scheme) and XMSSMT (XMSS Multi-Tree) [RFC8391] are stateful hash-based signature schemes [NIST800-208]. According to NIST, it is believed that the security of these schemes depends only on the security of the underlying hash functions, in particular the infeasibility of finding a preimage or a second preimage, and it is believed that the security of these hash functions will not be broken by the development of large-scale quantum computers.

For further information on the intended usage of these signature schemes and the careful state management required to maintain their strength, see [NIST800-208].

IANA maintains a registry whose entries correspond to the XMSS Identifiers below (see [XMSS]). The fragment part of the URIs is formed by replacing occurrences of underscore ("_") in the name appearing in the IANA registry with hyphen ("-").

Identifiers for XMSS:

- http://www.w3.org/2021/04/xmldsig-more#xmss-sha2-10-192
- http://www.w3.org/2021/04/xmldsig-more#xmss-sha2-10-256
- http://www.w3.org/2021/04/xmldsig-more#xmss-sha2-10-512
- http://www.w3.org/2021/04/xmldsig-more#xmss-sha2-16-192
- http://www.w3.org/2021/04/xmldsig-more#xmss-sha2-16-256
- http://www.w3.org/2021/04/xmldsig-more#xmss-sha2-16-512
- http://www.w3.org/2021/04/xmldsig-more#xmss-sha2-20-192
- http://www.w3.org/2021/04/xmldsig-more#xmss-sha2-20-256
- http://www.w3.org/2021/04/xmldsig-more#xmss-sha2-20-512
- http://www.w3.org/2021/04/xmldsig-more#xmss-shake-10-256
- http://www.w3.org/2021/04/xmldsig-more#xmss-shake-10-512
- http://www.w3.org/2021/04/xmldsig-more#xmss-shake-16-256
- http://www.w3.org/2021/04/xmldsig-more#xmss-shake-16-512
- http://www.w3.org/2021/04/xmldsig-more#xmss-shake-20-256
- http://www.w3.org/2021/04/xmldsig-more#xmss-shake-20-512

http://www.w3.org/2021/04/xmldsig-more#xmss-shake256-10-192
http://www.w3.org/2021/04/xmldsig-more#xmss-shake256-10-256
http://www.w3.org/2021/04/xmldsig-more#xmss-shake256-16-192
http://www.w3.org/2021/04/xmldsig-more#xmss-shake256-16-256
http://www.w3.org/2021/04/xmldsig-more#xmss-shake256-20-192
http://www.w3.org/2021/04/xmldsig-more#xmss-shake256-20-256

The hash functions used in the XMSS signature schemes above are SHA2 [RFC6234] or one of the two SHAKE extensible output functions [FIPS202] as indicated by the second token of the URI extension (SHAKE means SHAKE128). The tree height for XMSS is 10, 16, or 20 as indicated by the third token of the URI extension. The SHA2 or SHAKE output size is 192, 256, or 512 bits as indicated by the final token of the URI extension. SHA2 with 192 bits of output means SHA2-256/192, that is, the most significant 192 bits of the SHA-256 hash as specified in [NIST800-208].

IANA maintains a registry whose entries correspond to the XMSSMT Identifiers below (see [XMSS]). The fragment part of the URIs is formed by replacing occurrences of underscore ("_") and slash ("/") in the name appearing in the IANA registry with hyphen ("-").

Identifiers for XMSSMT:

http://www.w3.org/2021/04/xmldsig-more#xmssmt-sha2-20-2-192
http://www.w3.org/2021/04/xmldsig-more#xmssmt-sha2-20-2-256
http://www.w3.org/2021/04/xmldsig-more#xmssmt-sha2-20-2-512
http://www.w3.org/2021/04/xmldsig-more#xmssmt-sha2-20-4-192
http://www.w3.org/2021/04/xmldsig-more#xmssmt-sha2-20-4-256
http://www.w3.org/2021/04/xmldsig-more#xmssmt-sha2-20-4-512
http://www.w3.org/2021/04/xmldsig-more#xmssmt-sha2-40-2-192
http://www.w3.org/2021/04/xmldsig-more#xmssmt-sha2-40-2-256
http://www.w3.org/2021/04/xmldsig-more#xmssmt-sha2-40-2-512
http://www.w3.org/2021/04/xmldsig-more#xmssmt-sha2-40-4-192
http://www.w3.org/2021/04/xmldsig-more#xmssmt-sha2-40-4-256
http://www.w3.org/2021/04/xmldsig-more#xmssmt-sha2-40-4-512
http://www.w3.org/2021/04/xmldsig-more#xmssmt-sha2-40-8-192
http://www.w3.org/2021/04/xmldsig-more#xmssmt-sha2-40-8-256
http://www.w3.org/2021/04/xmldsig-more#xmssmt-sha2-40-8-512
http://www.w3.org/2021/04/xmldsig-more#xmssmt-sha2-60-3-192
http://www.w3.org/2021/04/xmldsig-more#xmssmt-sha2-60-3-256
http://www.w3.org/2021/04/xmldsig-more#xmssmt-sha2-60-3-512
http://www.w3.org/2021/04/xmldsig-more#xmssmt-sha2-60-6-192
http://www.w3.org/2021/04/xmldsig-more#xmssmt-sha2-60-6-256
http://www.w3.org/2021/04/xmldsig-more#xmssmt-sha2-60-6-512
http://www.w3.org/2021/04/xmldsig-more#xmssmt-sha2-60-12-192
http://www.w3.org/2021/04/xmldsig-more#xmssmt-sha2-60-12-256
http://www.w3.org/2021/04/xmldsig-more#xmssmt-sha2-60-12-512

http://www.w3.org/2021/04/xmldsig-more#xmssmt-shake-20-2-256
http://www.w3.org/2021/04/xmldsig-more#xmssmt-shake-20-2-512
http://www.w3.org/2021/04/xmldsig-more#xmssmt-shake-20-4-256
http://www.w3.org/2021/04/xmldsig-more#xmssmt-shake-20-4-512
http://www.w3.org/2021/04/xmldsig-more#xmssmt-shake-40-2-256
http://www.w3.org/2021/04/xmldsig-more#xmssmt-shake-40-2-512
http://www.w3.org/2021/04/xmldsig-more#xmssmt-shake-40-4-256
http://www.w3.org/2021/04/xmldsig-more#xmssmt-shake-40-4-512

http://www.w3.org/2021/04/xmldsig-more#xmssmt-shake-40-8-256
http://www.w3.org/2021/04/xmldsig-more#xmssmt-shake-40-8-512
http://www.w3.org/2021/04/xmldsig-more#xmssmt-shake-60-3-256
http://www.w3.org/2021/04/xmldsig-more#xmssmt-shake-60-3-512
http://www.w3.org/2021/04/xmldsig-more#xmssmt-shake-60-6-256
http://www.w3.org/2021/04/xmldsig-more#xmssmt-shake-60-6-512
http://www.w3.org/2021/04/xmldsig-more#xmssmt-shake-60-12-256
http://www.w3.org/2021/04/xmldsig-more#xmssmt-shake-60-12-512
http://www.w3.org/2021/04/xmldsig-more#xmssmt-shake256-20-2-192
http://www.w3.org/2021/04/xmldsig-more#xmssmt-shake256-20-2-256
http://www.w3.org/2021/04/xmldsig-more#xmssmt-shake256-20-4-192
http://www.w3.org/2021/04/xmldsig-more#xmssmt-shake256-20-4-256
http://www.w3.org/2021/04/xmldsig-more#xmssmt-shake256-40-2-192
http://www.w3.org/2021/04/xmldsig-more#xmssmt-shake256-40-2-256
http://www.w3.org/2021/04/xmldsig-more#xmssmt-shake256-40-4-192
http://www.w3.org/2021/04/xmldsig-more#xmssmt-shake256-40-4-256
http://www.w3.org/2021/04/xmldsig-more#xmssmt-shake256-40-8-192
http://www.w3.org/2021/04/xmldsig-more#xmssmt-shake256-40-8-256
http://www.w3.org/2021/04/xmldsig-more#xmssmt-shake256-60-3-192
http://www.w3.org/2021/04/xmldsig-more#xmssmt-shake256-60-3-256
http://www.w3.org/2021/04/xmldsig-more#xmssmt-shake256-60-6-192
http://www.w3.org/2021/04/xmldsig-more#xmssmt-shake256-60-6-256
http://www.w3.org/2021/04/xmldsig-more#xmssmt-shake256-60-12-192
http://www.w3.org/2021/04/xmldsig-more#xmssmt-shake256-60-12-256

The hash functions used in the XMSSMT signature schemes above are SHA2 [RFC6234] or one of the two the SHAKE extensible output function [FIPS202] as indicated by the second token of the URI extension (SHAKE means SHAKE128). The tree height for XMSSMT is 20, 40, or 60 as indicated by the third token of the URI extension. The number of layers is indicated by a fourth token. The SHA2, SHAKE, or SHAKE256 output size is 192, 256, or 512 bits as indicated by the final token of the URI extension. SHA2 with 192 bits of output means SHA2-256/192, that is, the most significant 192 bits of the SHA-256 hash as specified in [NIST800-208].

An example of an XMSS SignatureAlgorithm element is:

```
<SignatureAlgorithm
  Algorithm="http://www.w3.org/2021/04/xmldsig-more#xmss-sha2-10-192"
  />
```

2.3. SignatureMethod Public Key Signature Algorithms

These algorithms are distinguished from those in Section 2.2 in that they use public key methods. That is to say, the signing key is different from and not feasibly derivable from the verification 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 [RFC8017]. 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 [RFC4648] encoding of the octet string computed as per Section 8.2.1 of [RFC8017], signature generation for the RSASSA-PKCS1-v1_5 signature scheme. As specified in the EMSA-PKCS1-V1_5-ENCODE function in Section 9.2 of [RFC8017], the value input to the signature function MUST contain a prepended 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))))
```

The padded ASN.1 will be of the following form:

```
01 | FF* | 00 | prefix | hash
```

The 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 [RFC8017], 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 [RFC8017] 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 [RFC8017] 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 [RFC8017] 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 [RFC8017] 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/2021/04/xmldsig-more#ecdsa-sha3-224`
`http://www.w3.org/2021/04/xmldsig-more#ecdsa-sha3-256`
`http://www.w3.org/2021/04/xmldsig-more#ecdsa-sha3-384`
`http://www.w3.org/2021/04/xmldsig-more#ecdsa-sha3-512`
`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) [FIPS186-4] 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 some detailed specifications of how to use it with SHA hash functions and XML Digital Signature, please see [X9.62] and [RFC4050]. The #sha3-*, #ecdsa-ripemd160, and #ecdsa-whirlpool fragments identify signature methods processed in the same way as specified by the #ecdsa-sha1 fragment, with the exception that a SHA3 function (see Section 2.1.5), RIPEMD160, or Whirlpool (see Section 2.1.4) is used instead of SHA-1.

The output of the ECDSA algorithm consists of a pair of integers usually referred to as 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 [RFC8017] specification with the l parameter equal to the size of the base point order of the curve in octets (e.g., 32 for the P-256 curve and 66 for the P-521 curve [FIPS186-4]).

For an introduction to elliptic curve cryptographic algorithms, see [RFC6090] and note the errata (Errata IDs 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.

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 [RFC8017]. 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 [ISO-10118-3].

An example of use is:

```
<SignatureMethod
   Algorithm="http://www.w3.org/2007/05/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 use the PKCS #1 EMSA-PSS encoding algorithm [RFC8017]. The RSASSA-PSS algorithm takes the digest method (hash function), a mask generation function, the salt length in octets (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 [RFC8017], 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 [RFC8017], 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

[RFC8017] currently specifies only one mask generation function MGF1 based on a hash function. Although [RFC8017] 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 [RFC8017]. 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/2001/04/xmldsig-more#rsa-sha224
```

This implies the PKCS #1 v1.5 padding algorithm [RFC8017] 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/2001/04/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.

See also Appendix B concerning an erroneous version of this URI that appeared in [RFC6931].

2.3.12. Edwards-Curve

The Edwards-curve Digital Signature Algorithm (EdDSA) is a variant of Schnorr's signature system with Edwards curves. A specification is provided and some advantages listed in [RFC8032]. The general EdDSA takes 11 parameters that must be carefully chosen for secure and efficient operation. Identifiers for two variants, Ed25519 and Ed448, are given below.

Ed25519 uses 32-octet public keys and produces 64-octet signatures. It provides about 128 bits of security and uses SHA-512 [RFC6234] internally as part of signature generation.

Ed448 uses 57-octet public keys and produces 114-octet signatures. It provides about 224 bits of security and uses "SHAKE256" [FIPS202] internally as part of signature generation. (SHAKE256 is specified by NIST as an "Extensible Output Function" and not specified or approved by NIST as a secure hash function.)

For further information on the variants of EdDSA identified below, see [RFC8032].

Identifiers:

http://www.w3.org/2021/04/xmldsig-more#eddsa-ed25519ph
http://www.w3.org/2021/04/xmldsig-more#eddsa-ed25519ctx
http://www.w3.org/2021/04/xmldsig-more#eddsa-ed25519

http://www.w3.org/2021/04/xmldsig-more#eddsa-ed448
http://www.w3.org/2021/04/xmldsig-more#eddsa-ed448ph

An example of use is:

```
<SignatureMethod Algorithm=
   "http://www.w3.org/2021/04/xmldsig-more#eddsa-ed448" />
```

2.4. Minimal Canonicalization

Thus far, two independent interoperable implementations of Minimal Canonicalization have not been announced. Therefore, when "XML-Signature Syntax and Processing" was advanced along the Standards Track from [RFC3075] to [RFC3275], Minimal Canonicalization was dropped. However, there was 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

The XPointer Transform algorithm syntax is described below. 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] (Rivest Cipher 4); however, RC4 has been found to have a number of weaknesses and its use is prohibited in several IETF protocols, for example TLS [RFC7465]. An example EncryptionMethod element using ARCFOUR is:

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

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 regardless of 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, RSAES-KEM, and ECIES-KEM

Identifiers:
   http://www.w3.org/2001/04/xmldsig-more#psec-kem
   http://www.w3.org/2010/xmlsec-ghc#rsaes-kem
   http://www.w3.org/2010/xmlsec-ghc#ecies-kem

These algorithms, specified in [ISO-18033-2], are key encapsulation mechanisms using elliptic curve or RSA encryption. RSAEA-KEM and ECIES-KEM are also specified in [GENERIC].

An example of use of PSEC-KEM is:

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

See [ISO-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 block cipher with a 128-bit block size and 128-bit key size. 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"
/>
```

2.6.7. ChaCha20

Identifier:
http://www.w3.org/2021/04/xmldsig-more#chacha20

ChaCha20 [RFC8439], a stream cipher, is a variant of Salsa20 [ChaCha]. It is considerably faster than AES in software-only implementations. In addition to a 256-bit key and the plain text to be encrypted, ChaCha20 takes a 96-bit Nonce and an initial 32-bit

Counter. The Nonce and Counter are represented as hex in nested elements as shown below.

An example of use is:

```
<EncryptionMethod
   Algorithm=
   "http://www.w3.org/2021/04/xmldsig-more#chacha20">
   <Nonce>0123456789abcdef01234567</Nonce>
   <Counter>fedcba09</Counter>
</EncryptionMethod>
```

2.6.8. ChaCha20+Poly1305

Identifier:
http://www.w3.org/2021/04/xmldsig-more#chacha20poly1305

ChaCha20+Poly1305 is an Authenticated Encryption with Associated Data (AEAD) algorithm. In addition to a 256-bit key and plain text to be encrypted and authenticated, ChaCha20+Poly1305 takes a 96-bit Nonce and variable-length Additional Authenticated Data (AAD). The Nonce is represented as a child element of the EncryptionMethod element with a hex value. The AAD is a string, which may be null. The AAD element may be absent, in which case the AAD is null. The CipherData, either present in the CipherValue or by reference, is the concatenation of the encrypted ChaCha20 output and the Poly1305 128-bit tag.

An example of use is:

```
<EncryptionMethod
   Algorithm=
   "http://www.w3.org/2021/04/xmldsig-more#chacha20poly1305">
   <Nonce>0123456789abcdef01234567</Nonce>
   <AAD>The quick brown fox jumps over the lazy dog.</AAD>
</EncryptionMethod>
```

2.7. Key AgreementMethod Algorithm

This subsection gives identifiers and information for an additional key AgreementMethod Algorithm [XMLENC11].

2.7.1. X25519 and X448 Key Agreement

Identifier:
http://www.w3.org/2021/04/xmldsig-more#x25519
http://www.w3.org/2021/04/xmldsig-more#x448

The X25519 and X448 key agreement algorithms are specified in [RFC7748].

2.8. KeyDerivationMethod Algorithm

This subsection gives identifiers and information for an additional KeyDerivationMethod Algorithm [XMLENC11].

2.8.1. HKDF Key Derivation

This section covers the HMAC-based Extract-and-Expand Key Derivation Function (HKDF [RFC5869]).

Identifier:
http://www.w3.org/2021/04/xmldsig-more#hkdf

HKDF takes as inputs a hash function, an optional non-secret "salt", initial keying material (IKM), optional context and application-specific "info", and the required output keying size. Note that these strictly determine the output so, for example, invoking HKDF at different times but with the same salt, info, initial keying material, and output key size will produce identical output keying material.

The inputs can be supplied to HKDF as follows:

hash function: The algorithm attribute of a child DigestMethod element.

salt: The content of a Salt child element of AgreementMethod in hex. If not provided, a string of zero octets as long as the hash function output is used as specified in [RFC5869].

IKM: The content of an OriginatorKeyInfo child element of AgreementMethod in hex. May be absent in some applications where this is known through some other method.

info: The content of the KA-Nonce child element of AgreementMethod in hex.

size: The content of a KeySize child element of AgreementMethod as a decimal number.

Here is the test case from Appendix A.1 of [RFC5869] as an example:

```
<AgreementMethod
  algorithm="http://www.w3.org/2021/04/xmldsig-more#hkdf">
  <DigestMethod
    algorithm="http://www.w3.org/2001/04/xmldsig-more#hmac-sha256"/>
  <Salt>000102030405060708090a0b0c</Salt>
  <OriginatorKeyInfo>0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b
  </OriginatorKeyInfo>
  <KA-Nonce>f0f1f2f3f4f5f6f7f8f9</KA-Nonce>
  <KeySize>42</KeySize>
</AgreementMethod>
```

3. KeyInfo

In Section 3.1, a 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 [RFC4648] 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 standards document where the item is specified. See also [XMLSECXREF].

4.1. Index by 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. "{Bad}" indicates a bad value that was accidentally included in [RFC6931]. Implementations SHOULD only generate the correct URI but SHOULD understand both the correct and erroneous URI. See also Appendix B.

Fragment	URI	Sec/Doc
	2002/06/xmldsig-filter2	[XPATH]
	2006/12/xmlc12n11# {Bad}	[CANON11]
	2006/12/xmlc14n11#	[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
chacha20	2021/04/xmldsig-more#chacha20	2.6.7
chacha20poly1305	2021/04/xmldsig-more#chacha20poly1305	2.6.8
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-sha3-224	2021/04/xmldsig-more#ecdsa-sha3-224	2.3.6
ecdsa-sha3-256	2021/04/xmldsig-more#ecdsa-sha3-256	2.3.6
ecdsa-sha3-384	2021/04/xmldsig-more#ecdsa-sha3-384	2.3.6
ecdsa-sha3-512	2021/04/xmldsig-more#ecdsa-sha3-512	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]
eddsa-ed25519	2021/04/xmldsig-more#eddsa-ed25519	2.3.12
eddsa-ed25519ctx	2021/04/xmldsig-more#eddsa-ed25519ctx	2.3.12
eddsa-ed25519ph	2021/04/xmldsig-more#eddsa-ed25519ph	2.3.12
eddsa-ed448	2021/04/xmldsig-more#eddsa-ed448	2.3.12
eddsa-ed448ph	2021/04/xmldsig-more#eddsa-ed448ph	2.3.12
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]

hkdf                2021/04/xmldsig-more#hkdf              2.8.1
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
poly1305            2021/04/xmldsig-more#poly1305          2.2.4
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 {Bad}  2.3.11
rsa-sha224           2001/04/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
siphash-2-4          2021/04/xmldsig-more#siphash-2-4       2.2.5
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]

x25519               2021/04/xmldsig-more#x25519            2.7.1
x448                 2021/04/xmldsig-more#x448              2.7.1
X509Data             2000/09/xmldsig#X509Data          [XMLDSIG11]

xmss-sha2-10-192     2021/04/xmldsig-more#xmss-sha2-10-192  2.2.6
xmss-sha2-10-256     2021/04/xmldsig-more#xmss-sha2-10-256  2.2.6
xmss-sha2-10-512     2021/04/xmldsig-more#xmss-sha2-10-512  2.2.6
xmss-sha2-16-192     2021/04/xmldsig-more#xmss-sha2-16-192  2.2.6
```

```
xmss-sha2-16-256    2021/04/xmldsig-more#xmss-sha2-16-256  2.2.6
xmss-sha2-16-512    2021/04/xmldsig-more#xmss-sha2-16-512  2.2.6
xmss-sha2-20-192    2021/04/xmldsig-more#xmss-sha2-20-192  2.2.6
xmss-sha2-20-256    2021/04/xmldsig-more#xmss-sha2-20-256  2.2.6
xmss-sha2-20-512    2021/04/xmldsig-more#xmss-sha2-20-512  2.2.6
xmss-shake-10-256   2021/04/xmldsig-more#xmss-shake-10-256 2.2.6
xmss-shake-10-512   2021/04/xmldsig-more#xmss-shake-10-512 2.2.6
xmss-shake-16-256   2021/04/xmldsig-more#xmss-shake-16-256 2.2.6
xmss-shake-16-512   2021/04/xmldsig-more#xmss-shake-16-512 2.2.6
xmss-shake-20-256   2021/04/xmldsig-more#xmss-shake-20-256 2.2.6
xmss-shake-20-512   2021/04/xmldsig-more#xmss-shake-20-512 2.2.6
xmss-shake256-10-192 2021/04/xmldsig-more#xmss-shake256-10-192
                                                           2.2.6
xmss-shake256-10-256 2021/04/xmldsig-more#xmss-shake256-10-256
                                                           2.2.6
xmss-shake256-16-192 2021/04/xmldsig-more#xmss-shake256-16-192
                                                           2.2.6
xmss-shake256-16-256 2021/04/xmldsig-more#xmss-shake256-16-256
                                                           2.2.6
xmss-shake256-20-192 2021/04/xmldsig-more#xmss-shake256-20-192
                                                           2.2.6
xmss-shake256-20-256 2021/04/xmldsig-more#xmss-shake256-20-256
                                                           2.2.6
xmssmt-sha2-20-2-192 2021/04/xmldsig-more#xmssmt-sha2-20-2-192
                                                           2.2.6
xmssmt-sha2-20-2-256 2021/04/xmldsig-more#xmssmt-sha2-20-2-256
                                                           2.2.6
xmssmt-sha2-20-2-256 2021/04/xmldsig-more#xmssmt-sha2-20-2-512
                                                           2.2.6
xmssmt-sha2-20-4-192 2021/04/xmldsig-more#xmssmt-sha2-20-4-192
                                                           2.2.6
xmssmt-sha2-20-4-256 2021/04/xmldsig-more#xmssmt-sha2-20-4-256
                                                           2.2.6
xmssmt-sha2-20-4-256 2021/04/xmldsig-more#xmssmt-sha2-20-4-512
                                                           2.2.6
xmssmt-sha2-40-2-192 2021/04/xmldsig-more#xmssmt-sha2-40-2-192
                                                           2.2.6
xmssmt-sha2-40-2-256 2021/04/xmldsig-more#xmssmt-sha2-40-2-256
                                                           2.2.6
xmssmt-sha2-40-2-256 2021/04/xmldsig-more#xmssmt-sha2-40-2-512
                                                           2.2.6
xmssmt-sha2-40-4-192 2021/04/xmldsig-more#xmssmt-sha2-40-4-192
                                                           2.2.6
xmssmt-sha2-40-4-256 2021/04/xmldsig-more#xmssmt-sha2-40-4-256
                                                           2.2.6
xmssmt-sha2-40-4-256 2021/04/xmldsig-more#xmssmt-sha2-40-4-512
                                                           2.2.6
xmssmt-sha2-40-8-192 2021/04/xmldsig-more#xmssmt-sha2-40-8-192
                                                           2.2.6
xmssmt-sha2-40-8-256 2021/04/xmldsig-more#xmssmt-sha2-40-8-256
                                                           2.2.6
xmssmt-sha2-40-8-256 2021/04/xmldsig-more#xmssmt-sha2-40-8-512
                                                           2.2.6
xmssmt-sha2-60-3-192 2021/04/xmldsig-more#xmssmt-sha2-60-3-192
                                                           2.2.6
xmssmt-sha2-60-3-256 2021/04/xmldsig-more#xmssmt-sha2-60-3-256
```

2.2.6
xmssmt-sha2-60-3-256 2021/04/xmldsig-more#xmssmt-sha2-60-3-512
2.2.6
xmssmt-sha2-60-6-192 2021/04/xmldsig-more#xmssmt-sha2-60-6-192
2.2.6
xmssmt-sha2-60-6-256 2021/04/xmldsig-more#xmssmt-sha2-60-6-256
2.2.6
xmssmt-sha2-60-6-256 2021/04/xmldsig-more#xmssmt-sha2-60-6-512
2.2.6
xmssmt-sha2-60-12-192 2021/04/xmldsig-more#xmssmt-sha2-60-12-192
2.2.6
xmssmt-sha2-60-12-256 2021/04/xmldsig-more#xmssmt-sha2-60-12-256
2.2.6
xmssmt-sha2-60-12-256 2021/04/xmldsig-more#xmssmt-sha2-60-12-512
2.2.6
xmssmt-shake-20-2-256 2021/04/xmldsig-more#xmssmt-shake-20-2-256
2.2.6
xmssmt-shake-20-2-512 2021/04/xmldsig-more#xmssmt-shake-20-2-512
2.2.6
xmssmt-shake-20-4-256 2021/04/xmldsig-more#xmssmt-shake-20-4-256
2.2.6
xmssmt-shake-20-4-512 2021/04/xmldsig-more#xmssmt-shake-20-4-512
2.2.6
xmssmt-shake-40-2-256 2021/04/xmldsig-more#xmssmt-shake-40-2-256
2.2.6
xmssmt-shake-40-2-512 2021/04/xmldsig-more#xmssmt-shake-40-2-512
2.2.6
xmssmt-shake-40-4-256 2021/04/xmldsig-more#xmssmt-shake-40-4-256
2.2.6
xmssmt-shake-40-4-512 2021/04/xmldsig-more#xmssmt-shake-40-4-512
2.2.6
xmssmt-shake-40-8-256 2021/04/xmldsig-more#xmssmt-shake-40-8-256
2.2.6
xmssmt-shake-40-8-512 2021/04/xmldsig-more#xmssmt-shake-40-8-512
2.2.6
xmssmt-shake-60-3-256 2021/04/xmldsig-more#xmssmt-shake-60-3-256
2.2.6
xmssmt-shake-60-3-512 2021/04/xmldsig-more#xmssmt-shake-60-3-512
2.2.6
xmssmt-shake-60-6-256 2021/04/xmldsig-more#xmssmt-shake-60-6-256
2.2.6
xmssmt-shake-60-6-512 2021/04/xmldsig-more#xmssmt-shake-60-6-512
2.2.6
xmssmt-shake-60-12-256 2021/04/xmldsig-more#xmssmt-shake-20-12-256
2.2.6
xmssmt-shake-60-12-512 2021/04/xmldsig-more#xmssmt-shake-20-12-512
2.2.6

xmssmt-shake256-20-2-192
2021/04/xmldsig-more#xmssmt-shake256-20-2-192 2.2.6
xmssmt-shake256-20-2-256
2021/04/xmldsig-more#xmssmt-shake256-20-2-256 2.2.6
xmssmt-shake256-20-4-192
2021/04/xmldsig-more#xmssmt-shake256-20-4-192 2.2.6
xmssmt-shake256-20-4-256
2021/04/xmldsig-more#xmssmt-shake256-20-4-256 2.2.6

xmssmt-shake256-40-2-192
2021/04/xmldsig-more#xmssmt-shake256-40-2-192 2.2.6
xmssmt-shake256-40-2-256
2021/04/xmldsig-more#xmssmt-shake256-40-2-256 2.2.6
xmssmt-shake256-40-4-192
2021/04/xmldsig-more#xmssmt-shake256-40-4-192 2.2.6
xmssmt-shake256-40-4-256
2021/04/xmldsig-more#xmssmt-shake256-40-4-256 2.2.6
xmssmt-shake256-40-8-192
2021/04/xmldsig-more#xmssmt-shake256-40-8-192 2.2.6
xmssmt-shake256-40-8-256
2021/04/xmldsig-more#xmssmt-shake256-40-8-256 2.2.6
xmssmt-shake256-60-3-192
2021/04/xmldsig-more#xmssmt-shake256-60-3-192 2.2.6
xmssmt-shake256-60-3-256
2021/04/xmldsig-more#xmssmt-shake256-60-3-256 2.2.6
xmssmt-shake256-60-6-192
2021/04/xmldsig-more#xmssmt-shake256-60-6-192 2.2.6
xmssmt-shake256-60-6-256
2021/04/xmldsig-more#xmssmt-shake256-60-6-256 2.2.6
xmssmt-shake256-60-12-192
2021/04/xmldsig-more#xmssmt-shake256-60-12-192 2.2.6
xmssmt-shake256-60-12-256
2021/04/xmldsig-more#xmssmt-shake256-60-12-256 2.2.6

xptr	2001/04/xmldsig-more#xptr	2.5.1
Fragment	URI	Sec/Doc

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

4.2. Index by URI

The initial "http://www.w3.org/" part of the URI is not included below. "{Bad}" indicates a Bad value that was accidentally included in [RFC6931]. Implementations SHOULD only generate the correct URI but SHOULD understand both the correct and erroneous URI. See also Appendix B.

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-sha224	2.3.11	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# {Bad}              [CANON11]    Canonicalization
2006/12/xmlc14n11#                    [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 {Bad}  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]    KeyDerivation
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]	KeyDerivation
2009/xmlenc11#rsa-oaep	[XMLENC11]	EncryptionMethod
2009/xmlenc11#ECDH-ES	[XMLENC11]	AgreementMethod
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
2021/04/xmldsig-more#chacha20	2.6.7	EncryptionMethod
2021/04/xmldsig-more#chacha20poly1305	2.6.8	EncryptionMethod
2021/04/xmldsig-more#ecdsa-sha3-224	2.3.6	SignatureMethod
2021/04/xmldsig-more#ecdsa-sha3-256	2.3.6	SignatureMethod
2021/04/xmldsig-more#ecdsa-sha3-384	2.3.6	SignatureMethod
2021/04/xmldsig-more#ecdsa-sha3-512	2.3.6	SignatureMethod
2021/04/xmldsig-more#eddsa-ed25519ph	2.3.12	SignatureMethod
2021/04/xmldsig-more#eddsa-ed25519ctx	2.3.12	SignatureMethod
2021/04/xmldsig-more#eddsa-ed25519	2.3.12	SignatureMethod
2021/04/xmldsig-more#eddsa-ed448	2.3.12	SignatureMethod
2021/04/xmldsig-more#eddsa-ed448ph	2.3.12	SignatureMethod
2021/04/xmldsig-more#hkdf	2.8.1	KeyDerivation
2021/04/xmldsig-more#po1y305	2.2.4	SignatureMethod
2021/04/xmldsig-more#siphash-2-4	2.2.5	SignatureMethod
2021/04/xmldsig-more#x25519	2.7.1	AgreementMethod
2021/04/xmldsig-more#x448	2.7.1	AgreementMethod
2021/04/xmldsig-more#xmss-sha2-10-192	2.2.6	SignatureMethod
2021/04/xmldsig-more#xmss-sha2-10-256	2.2.6	SignatureMethod
2021/04/xmldsig-more#xmss-sha2-10-512	2.2.6	SignatureMethod
2021/04/xmldsig-more#xmss-sha2-16-192	2.2.6	SignatureMethod
2021/04/xmldsig-more#xmss-sha2-16-256	2.2.6	SignatureMethod
2021/04/xmldsig-more#xmss-sha2-16-512	2.2.6	SignatureMethod
2021/04/xmldsig-more#xmss-sha2-20-192	2.2.6	SignatureMethod
2021/04/xmldsig-more#xmss-sha2-20-256	2.2.6	SignatureMethod
2021/04/xmldsig-more#xmss-sha2-20-512	2.2.6	SignatureMethod
2021/04/xmldsig-more#xmss-shake-10-256	2.2.6	SignatureMethod
2021/04/xmldsig-more#xmss-shake-10-512	2.2.6	SignatureMethod
2021/04/xmldsig-more#xmss-shake-16-256	2.2.6	SignatureMethod
2021/04/xmldsig-more#xmss-shake-16-512	2.2.6	SignatureMethod
2021/04/xmldsig-more#xmss-shake-20-256	2.2.6	SignatureMethod
2021/04/xmldsig-more#xmss-shake-20-512	2.2.6	SignatureMethod
2021/04/xmldsig-more#xmss-shake256-10-192	2.2.6	SignatureMethod
2021/04/xmldsig-more#xmss-shake256-10-256	2.2.6	SignatureMethod
2021/04/xmldsig-more#xmss-shake256-16-192	2.2.6	SignatureMethod
2021/04/xmldsig-more#xmss-shake256-16-256	2.2.6	SignatureMethod
2021/04/xmldsig-more#xmss-shake256-20-192	2.2.6	SignatureMethod
2021/04/xmldsig-more#xmss-shake256-20-256	2.2.6	SignatureMethod
2021/04/xmldsig-more#xmssmt-sha2-20-2-192	2.2.6	SignatureMethod
2021/04/xmldsig-more#xmssmt-sha2-20-2-256	2.2.6	SignatureMethod
2021/04/xmldsig-more#xmssmt-sha2-20-2-512	2.2.6	SignatureMethod
2021/04/xmldsig-more#xmssmt-sha2-20-4-192	2.2.6	SignatureMethod
2021/04/xmldsig-more#xmssmt-sha2-20-4-256	2.2.6	SignatureMethod
2021/04/xmldsig-more#xmssmt-sha2-20-4-512	2.2.6	SignatureMethod
2021/04/xmldsig-more#xmssmt-sha2-40-2-192	2.2.6	SignatureMethod
2021/04/xmldsig-more#xmssmt-sha2-40-2-256	2.2.6	SignatureMethod

2021/04/xmldsig-more#xmssmt-sha2-40-2-512 2.2.6 SignatureMethod
2021/04/xmldsig-more#xmssmt-sha2-40-4-192 2.2.6 SignatureMethod
2021/04/xmldsig-more#xmssmt-sha2-40-4-256 2.2.6 SignatureMethod
2021/04/xmldsig-more#xmssmt-sha2-40-4-512 2.2.6 SignatureMethod
2021/04/xmldsig-more#xmssmt-sha2-40-8-192 2.2.6 SignatureMethod
2021/04/xmldsig-more#xmssmt-sha2-40-8-256 2.2.6 SignatureMethod
2021/04/xmldsig-more#xmssmt-sha2-40-8-512 2.2.6 SignatureMethod
2021/04/xmldsig-more#xmssmt-sha2-60-3-192 2.2.6 SignatureMethod
2021/04/xmldsig-more#xmssmt-sha2-60-3-256 2.2.6 SignatureMethod
2021/04/xmldsig-more#xmssmt-sha2-60-3-512 2.2.6 SignatureMethod
2021/04/xmldsig-more#xmssmt-sha2-60-6-192 2.2.6 SignatureMethod
2021/04/xmldsig-more#xmssmt-sha2-60-6-256 2.2.6 SignatureMethod
2021/04/xmldsig-more#xmssmt-sha2-60-6-512 2.2.6 SignatureMethod
2021/04/xmldsig-more#xmssmt-sha2-60-12-192 2.2.6 SignatureMethod
2021/04/xmldsig-more#xmssmt-sha2-60-12-256 2.2.6 SignatureMethod
2021/04/xmldsig-more#xmssmt-sha2-60-12-512 2.2.6 SignatureMethod

2021/04/xmldsig-more#xmssmt-shake-20-2-256 2.2.6 SignatureMethod
2021/04/xmldsig-more#xmssmt-shake-20-2-512 2.2.6 SignatureMethod
2021/04/xmldsig-more#xmssmt-shake-20-4-256 2.2.6 SignatureMethod
2021/04/xmldsig-more#xmssmt-shake-20-4-512 2.2.6 SignatureMethod
2021/04/xmldsig-more#xmssmt-shake-40-2-256 2.2.6 SignatureMethod
2021/04/xmldsig-more#xmssmt-shake-40-2-512 2.2.6 SignatureMethod
2021/04/xmldsig-more#xmssmt-shake-40-4-256 2.2.6 SignatureMethod
2021/04/xmldsig-more#xmssmt-shake-40-4-512 2.2.6 SignatureMethod
2021/04/xmldsig-more#xmssmt-shake-40-8-256 2.2.6 SignatureMethod
2021/04/xmldsig-more#xmssmt-shake-40-8-512 2.2.6 SignatureMethod
2021/04/xmldsig-more#xmssmt-shake-60-3-256 2.2.6 SignatureMethod
2021/04/xmldsig-more#xmssmt-shake-60-3-512 2.2.6 SignatureMethod
2021/04/xmldsig-more#xmssmt-shake-60-6-256 2.2.6 SignatureMethod
2021/04/xmldsig-more#xmssmt-shake-60-6-512 2.2.6 SignatureMethod
2021/04/xmldsig-more#xmssmt-shake-60-12-256 2.2.6 SignatureMethod
2021/04/xmldsig-more#xmssmt-shake-60-12-512 2.2.6 SignatureMethod

2021/04/xmldsig-more#xmssmt-shake256-20-2-192
                                        2.2.6   SignatureMethod
2021/04/xmldsig-more#xmssmt-shake256-20-2-256
                                        2.2.6   SignatureMethod
2021/04/xmldsig-more#xmssmt-shake256-20-4-192
                                        2.2.6   SignatureMethod
2021/04/xmldsig-more#xmssmt-shake256-20-4-256
                                        2.2.6   SignatureMethod
2021/04/xmldsig-more#xmssmt-shake256-40-2-192
                                        2.2.6   SignatureMethod
2021/04/xmldsig-more#xmssmt-shake256-40-2-256
                                        2.2.6   SignatureMethod
2021/04/xmldsig-more#xmssmt-shake256-40-4-192
                                        2.2.6   SignatureMethod
2021/04/xmldsig-more#xmssmt-shake256-40-4-256
                                        2.2.6   SignatureMethod
2021/04/xmldsig-more#xmssmt-shake256-40-8-192
                                        2.2.6   SignatureMethod
2021/04/xmldsig-more#xmssmt-shake256-40-8-256
                                        2.2.6   SignatureMethod
2021/04/xmldsig-more#xmssmt-shake256-60-3-192
                                        2.2.6   SignatureMethod

```
2021/04/xmldsig-more#xmssmt-shake256-60-3-256
                                          2.2.6    SignatureMethod
2021/04/xmldsig-more#xmssmt-shake256-60-6-192
                                          2.2.6    SignatureMethod
2021/04/xmldsig-more#xmssmt-shake256-60-6-256
                                          2.2.6    SignatureMethod
2021/04/xmldsig-more#xmssmt-shake256-60-12-192
                                          2.2.6    SignatureMethod
2021/04/xmldsig-more#xmssmt-shake256-60-12-256
                                          2.2.6    SignatureMethod

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
----                                 --------    ------
 URI                                  Sec/Doc      Type
```

The initial "http://www.w3.org/" part of the URI is not included above. "{Bad}" indicates a Bad value that was accidentally included in [RFC6931]. Implementations SHOULD only generate the correct URI but SHOULD understand both the correct and erroneous URI. See also Appendix B.

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, additional URI specification under the following XMLSEC URI prefixes is prohibited as shown:

URI	Status
http://www.w3.org/2000/09/xmldsig#	Frozen by W3C.
http://www.w3.org/2001/04/xmldsig-more#	Frozen with RFC 4051.
http://www.w3.org/2007/05/xmldsig-more#	Frozen with [RFC6931].

Table 2

The W3C has assigned <http://www.w3.org/2021/04/xmldsig-more#> for additional new URIs specified in this document.

There are also occurrences in this document of <http://www.w3.org/2010/xmlsec-ghc#> due to the inclusion of some

algorithms from [GENERIC] for convenience.

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 contents have been updated to 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 "[RFC9231], Section 2.6.4"). All references to [RFC6931] in that registry have been updated to RFC 9231.

New entries, including new Types, will be added based on Specification Required [RFC8126]. Criteria for the designated expert 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 RFC.

Typically, the registry will reference a W3C or IETF document specifying such XML syntax; that document will either contain a more detailed description of the algorithm or data type or reference another document with a more detailed 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 the use of HMAC-MD5 or RSA-MD5 as a SignatureMethod.

See [RFC6194] for SHA-1 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 decrease. 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 for any particular use to change over time. This is sometimes referred to as "algorithm agility" [RFC7696].

7. References

7.1. Normative References

[FIPS180-4]
National Institute of Standards and Technology (NIST), "Secure Hash Standard (SHS)", DOI 10.6028/NIST.FIPS.180-4, FIPS 180-4, August 2015, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf>.

[FIPS186-4]
National Institute of Standards and Technology (NIST), "Digital Signature Standard (DSS)", FIPS 186-4, DOI 10.6028/NIST.FIPS.186-4, July 2013, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf>.

[FIPS202]
National Institute of Standards and Technology (NIST), "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions", FIPS 202, DOI 10.6028/NIST.FIPS.202, August 2015, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf>.

[IEEEP1363a]
Institute of Electrical and Electronics Engineers, "IEEE Standard Specifications for Public-Key Cryptography - Amendment 1: Additional Techniques", IEEE Std 1363a-2004, 2004.

[ISO-10118-3]
ISO, "Information technology -- Security techniques -- Hash-functions -- Part 3: Dedicated hash-functions", ISO/IEC 10118-3:2004, 2004.

[ISO-18033-2]
ISO, "Information technology -- Security techniques --Encryption algorithms -- Part 3: Asymmetric ciphers", ISO/IEC 18033-2:2010, 2010.

[NIST800-208]
National Institute of Standards and Technology (NIST), "Recommendation for Stateful Hash-Based Signature Schemes", NIST 800-208, DOI 10.6028/NIST.SP.800-208, October 2020, <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-208.pdf>.

[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, DOI 10.17487/RFC1321, April 1992, <https://www.rfc-editor.org/info/rfc1321>.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/info/rfc2104>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2315] Kaliski, B., "PKCS #7: Cryptographic Message Syntax Version 1.5", RFC 2315, DOI 10.17487/RFC2315, March 1998, <https://www.rfc-editor.org/info/rfc2315>.

[RFC3275] Eastlake 3rd, D., Reagle, J., and D. Solo, "(Extensible Markup Language) XML-Signature Syntax and Processing", RFC 3275, DOI 10.17487/RFC3275, March 2002, <https://www.rfc-editor.org/info/rfc3275>.

[RFC3394] Schaad, J. and R. Housley, "Advanced Encryption Standard (AES) Key Wrap Algorithm", RFC 3394, DOI 10.17487/RFC3394, September 2002, <https://www.rfc-editor.org/info/rfc3394>.

[RFC3713] Matsui, M., Nakajima, J., and S. Moriai, "A Description of the Camellia Encryption Algorithm", RFC 3713, DOI 10.17487/RFC3713, April 2004, <https://www.rfc-editor.org/info/rfc3713>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC4050] Blake-Wilson, S., Karlinger, G., Kobayashi, T., and Y. Wang, "Using the Elliptic Curve Signature Algorithm (ECDSA) for XML Digital Signatures", RFC 4050, DOI 10.17487/RFC4050, April 2005, <https://www.rfc-editor.org/info/rfc4050>.

[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, DOI 10.17487/RFC4055, June 2005, <https://www.rfc-editor.org/info/rfc4055>.

[RFC4269] Lee, H.J., Lee, S.J., Yoon, J.H., Cheon, D.H., and J.I. Lee, "The SEED Encryption Algorithm", RFC 4269, DOI 10.17487/RFC4269, December 2005, <https://www.rfc-editor.org/info/rfc4269>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <https://www.rfc-editor.org/info/rfc5869>.

[RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <https://www.rfc-editor.org/info/rfc6234>.

[RFC7748] Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves for Security", RFC 7748, DOI 10.17487/RFC7748, January 2016, <https://www.rfc-editor.org/info/rfc7748>.

[RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016, <https://www.rfc-editor.org/info/rfc8017>.

[RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <https://www.rfc-editor.org/info/rfc8032>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8391] Huelsing, A., Butin, D., Gazdag, S., Rijneveld, J., and A. Mohaisen, "XMSS: eXtended Merkle Signature Scheme", RFC 8391, DOI 10.17487/RFC8391, May 2018, <https://www.rfc-editor.org/info/rfc8391>.

[RFC8439] Nir, Y. and A. Langley, "ChaCha20 and Poly1305 for IETF Protocols", RFC 8439, DOI 10.17487/RFC8439, June 2018, <https://www.rfc-editor.org/info/rfc8439>.

[SipHash1] Aumasson, J. and D. Bernstein, "SipHash: A Fast Short-Input PRF", Progress in Cryptology - INDOCRYPT 2012, Lecture Notes in Computer Science vol. 7668, December 2012, <https://doi.org/10.1007/978-3-642-34931-7_28>.

[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 3rd, "XML Encryption Syntax and Processing", W3C Recommendation, December 2002, <https://www.w3.org/TR/2002/REC-xmlenc-core-20021210/>.

[XMLENC11] Eastlake 3rd, D., Reagle, J., Hirsch, F., and T. Roessler, "XML Encryption Syntax and Processing Version 1.1", W3C Proposed Recommendation, April 2013, <https://www.w3.org/TR/xmlenc-core1/>.

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

7.2. Informative References

[CAMELLIA] Aoki, K., Ichikawa, T., Kanda, M., 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.

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

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

[ChaCha] Bernstein, D., "ChaCha, a variant of Salsa20", January 2008, <https://cr.yp.to/chacha/chacha-20080128.pdf>.

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

[Err3597] RFC Errata, "Erratum ID 3597", RFC 6931, <https://www.rfc-editor.org/errata/eid3597>.

[Err3965] RFC Errata, "Erratum ID 3965", RFC 6931, <https://www.rfc-editor.org/errata/eid3965>.

[Err4004] RFC Errata, "Erratum ID 4004", RFC 6931, <https://www.rfc-editor.org/errata/eid4004>.

[GENERIC] Nyström, M. and F. Hirsch, "XML Security Generic Hybrid Ciphers", W3C Working Group Note, April 2013, <https://www.w3.org/TR/xmlsec-generic-hybrid/>.

[ITU-T-X.660] ITU-T, "Information technology - Procedures for the operation of object identifier registration authorities: General procedures and top arcs of the international object identifier tree", ITU-T Recommendation X.660, July 2011, <https://www.itu.int/rec/T-REC-X.660>.

[ITU-T-X.680] ITU-T, "Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation", ITU-T

Recommendation X.680, February 2021,
<https://www.itu.int/rec/T-REC-X.680>.

[KECCAK] Bertoni, G., Daeman, J., Peeters, M., and G. Van Assche, "KECCAK sponge function family", Version 2.1, June 2010, <https://keccak.team/obsolete/Keccak-main-2.1.pdf>.

[POLY1305] Bernstein, D., "The Poly1305-AES message-authentication code", March 2005, <https://cr.yp.to/mac/poly1305-20050329.pdf>.

[RFC3075] Eastlake 3rd, D., Reagle, J., and D. Solo, "XML-Signature Syntax and Processing", RFC 3075, DOI 10.17487/RFC3075, March 2001, <https://www.rfc-editor.org/info/rfc3075>.

[RFC3076] Boyer, J., "Canonical XML Version 1.0", RFC 3076, DOI 10.17487/RFC3076, March 2001, <https://www.rfc-editor.org/info/rfc3076>.

[RFC3092] Eastlake 3rd, D., Manros, C., and E. Raymond, "Etymology of "Foo"", RFC 3092, DOI 10.17487/RFC3092, April 2001, <https://www.rfc-editor.org/info/rfc3092>.

[RFC3741] Boyer, J., Eastlake 3rd, D., and J. Reagle, "Exclusive XML Canonicalization, Version 1.0", RFC 3741, DOI 10.17487/RFC3741, March 2004, <https://www.rfc-editor.org/info/rfc3741>.

[RFC4010] Park, J., Lee, S., Kim, J., and J. Lee, "Use of the SEED Encryption Algorithm in Cryptographic Message Syntax (CMS)", RFC 4010, DOI 10.17487/RFC4010, February 2005, <https://www.rfc-editor.org/info/rfc4010>.

[RFC6090] McGrew, D., Igoe, K., and M. Salter, "Fundamental Elliptic Curve Cryptography Algorithms", RFC 6090, DOI 10.17487/RFC6090, February 2011, <https://www.rfc-editor.org/info/rfc6090>.

[RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, DOI 10.17487/RFC6151, March 2011, <https://www.rfc-editor.org/info/rfc6151>.

[RFC6194] Polk, T., Chen, L., Turner, S., and P. Hoffman, "Security Considerations for the SHA-0 and SHA-1 Message-Digest Algorithms", RFC 6194, DOI 10.17487/RFC6194, March 2011, <https://www.rfc-editor.org/info/rfc6194>.

[RFC6931] Eastlake 3rd, D., "Additional XML Security Uniform Resource Identifiers (URIs)", RFC 6931, DOI 10.17487/RFC6931, April 2013, <https://www.rfc-editor.org/info/rfc6931>.

[RFC7465] Popov, A., "Prohibiting RC4 Cipher Suites", RFC 7465, DOI 10.17487/RFC7465, February 2015, <https://www.rfc-editor.org/info/rfc7465>.

[RFC7696] Housley, R., "Guidelines for Cryptographic Algorithm Agility and Selecting Mandatory-to-Implement Algorithms", BCP 201, RFC 7696, DOI 10.17487/RFC7696, November 2015, <https://www.rfc-editor.org/info/rfc7696>.

[SCHEMA] Thompson, H., Beech, D., Maloney, M., and N. Mendelsohn, "XML Schema Part 1: Structures Second Edition", W3C Recommendation REC-xmlschema-1-20041028, 28 October 2004.

Biron, P. and A. Malhotra, "XML Schema Part 2: Datatypes Second Edition", W3C Recommendation REC-xmlschema-2-20041028, 28 October 2004.

[SipHash2] Aumasson, J. and D. Bernstein, "SipHash: A Fast Short-Input PRF", Department of Computer Science, University of Illinois at Chicago, <https://www.aumasson.jp/siphash/siphash.pdf>.

[W3C] "World Wide Web Consortium (W3C)", <https://www.w3.org>.

[XCANON] Boyer, J., Eastlake 3rd, D., and J. Reagle, "Exclusive XML Canonicalization Version 1.0", W3C Recommendation, July 2002, <https://www.w3.org/TR/2002/REC-xml-exc-c14n-20020718/>.

[XMLDSIG-PROP] Hirsch, F., "XML Signature Properties", W3C Recommendation, April 2013, <https://www.w3.org/TR/xmldsig-properties/>.

[XMLDSIG10] Bartel, M., Boyer, J., Fox, B., Simon, E., and B. LaMacchia, "XML Signature Syntax and Processing (Second Edition)", W3C Recommendation, June 2008, <https://www.w3.org/TR/2008/REC-xmldsig-core-20080610/>.

[XMLDSIG11] Bartel, M., Boyer, J., Fox, B., Simon, E., and B. LaMacchia, "XML Signature Syntax and Processing Version 1.1", W3C Proposed Recommendation, April 2013, <https://www.w3.org/TR/xmldsig-core1/>.

[XMLSEC] Eastlake 3rd, D. and K. Niles, "Secure XML: The New Syntax for Signatures and Encryption", Addison-Wesley (Pearson Education) ISBN 0-201-75605-6, 2003.

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

[XMSS] IANA, "XMSS: Extended Hash-Based Signatures", <https://www.iana.org/assignments/xmss-extended-hash-based-signatures>.

[XPATH] Boyer, J., Hughes, M., and J. Reagle, "XML-Signature XPath Filter 2.0", W3C Recommendation REC-xmldsig-filter2-20021108, 8 November 2002.

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 REC-xpath20-20101214, 14 December 2010.

[XSLT] Kay, M., "XSL Transformations (XSLT) Version 2.0", W3C Recommendation, Second Edition, March 2021, <https://www.w3.org/TR/xslt20/>.

Appendix A. Changes from RFC 6931

The following changes have been made in [RFC6931] to produce this document.

* Deleted Appendix on Changes from RFC 4051, since they were already included in [RFC6931], and remove reference to RFC 4051 and to the one Errata against RFC 4051.

* Fixed three errata as follows: [Err3597], [Err3965], and [Err4004]. In cases where [RFC6931] had an erroneous URI, it is still included in the indices and it is stated that implementations SHOULD only generate the correct URI but SHOULD understand both the correct and erroneous URI.

* Added the following algorithms:

Section	Algorithm(s)
2.2.4	Poly1305
2.2.5	SipHash-2-4
2.2.6	XMSS and XMSSMT
2.3.6	ECDSA with SHA3
2.3.12	Edwards-Curve Signatures
2.6.7	ChaCha20
2.6.8	ChaCha20+Poly1305
2.7.1	X25519
2.8.1	HKDF

Table 3

* Listed ECIES-KEM and RSAES-KEM in Section 2.6.4 so they are easier to find even though the URI for them is specified in [GENERIC].

* Updated references for [GENERIC] and FIPS 186, added appropriate references.

* Added some XML examples.

* Fixed minor typos and added editorial changes.

* A number of acronyms were added to Section 1.2.

Appendix B. Bad URIs

[RFC6931] included two bad URIs as shown below. "{Bad}" in the indexes (Sections 4.1 and 4.2) indicates such a bad value. Implementations SHOULD only generate the correct URI but SHOULD understand both the correct and erroneous URI.

2006/12/xmlc12n11#

Appears in the indices (Sections 4.1 and 4.2 of [RFC6931]) when it should be "2006/12/xmlc14n11#" (i.e., the "12" inside "xmlc12n11" should have been "14"). This is [Err3965] and is corrected in this document.

2007/05/xmldsig-more#rsa-sha224

Appears in the indices (Sections 4.1 and 4.2 of [RFC6931]) when it should be "2001/04/xmldsig-more#rsa-sha224". This is [Err4004] and is corrected in this document.

Acknowledgements

The contributions of the following, listed in alphabetic order, by reporting errata against [RFC6931] or contributing to this document, are gratefully acknowledged:

Roman Danyliw, Pim van der Eijk, Frederick Hirsch, Benjamin Kaduk, Alexey Melnikov, Gayle Noble, Axel Puhlmann, Juraj Somorovsky, Peter Yee, and Annie Yousar.

The contributions of the following, listed in alphabetic order, to [RFC6931], on which this document is based, are gratefully acknowledged:

Benoit Claise, Adrian Farrel, Stephen Farrell, Ernst Giessmann, Frederick Hirsch, Björn Höhrmann, 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 RFC 4051 are gratefully acknowledged:

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

Author's Address

Donald E. Eastlake 3rd
Futurewei Technologies, Inc.
2386 Panoramic Circle
Apopka, FL 32703
United States of America
Phone: +1-508-333-2270
Email: d3e3e3@gmail.com