Stream: Internet Engineering Task Force (IETF)
RFC: 9688
Category: Standards Track
Published: November 2024
ISSN: 2070-1721
Author: R. Housley
Vigil Security

RFC 9688
Use of the SHA3 One-Way Hash Functions in the Cryptographic Message Syntax (CMS)

Abstract

This document describes the conventions for using the one-way hash functions in the SHA3 family with the Cryptographic Message Syntax (CMS). The SHA3 family can be used as a message digest algorithm, as part of a signature algorithm, as part of a message authentication code, or as part of a Key Derivation Function (KDF).

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/rfc9688.

Copyright Notice

Table of Contents

1. Introduction

The Cryptographic Message Syntax (CMS) [RFC5652] is used to digitally sign, digest, authenticate, or encrypt arbitrary message contents. This specification describes the use of the four one-way hash functions in the SHA3 family (SHA3-224, SHA3-256, SHA3-384, and SHA3-512) [SHA3] with the CMS. In addition, this specification describes the use of these four one-way hash functions with the RSASSA PKCS#1 version 1.5 signature algorithm [RFC8017] and the Elliptic Curve Digital Signature Algorithm (ECDSA) [DSS] with the CMS signed-data content type.

This document should not be confused with [RFC8702], which defines conventions for using the SHAKE family of SHA3-based extensible output functions with the CMS.

1.1. ASN.1

CMS values are generated using ASN.1 [X.680], using the Basic Encoding Rules (BER) and the Distinguished Encoding Rules (DER) [X.690].

1.2. 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.

2. Message Digest Algorithms

One-way hash functions are also referred to as message digest algorithms. This section specifies the conventions employed by CMS implementations that support SHA3-224, SHA3-256, SHA3-384, and SHA3-512 [SHA3].

Digest algorithm identifiers are located in the SignedData digestAlgorithms field, the SignerInfo digestAlgorithm field, the DigestedData digestAlgorithm field, and the AuthenticatedData digestAlgorithm field.

Digest values are located in the DigestedData digest field and the Message Digest authenticated attribute. In addition, digest values are input to signature algorithms.

SHA3-224, SHA3-256, SHA3-384, and SHA3-512 produce output values with 224, 256, 384, and 512 bits, respectively. The object identifiers for these four one-way hash functions are as follows:

```
hashAlgs OBJECT IDENTIFIER ::= { joint-iso-itu-t(2) country(16)
    us(840) organization(1) gov(101) csor(3) nistAlgorithm(4) 2 }

id-sha3-224 OBJECT IDENTIFIER ::= { hashAlgs 7 }

id-sha3-256 OBJECT IDENTIFIER ::= { hashAlgs 8 }

id-sha3-384 OBJECT IDENTIFIER ::= { hashAlgs 9 }

id-sha3-512 OBJECT IDENTIFIER ::= { hashAlgs 10 }
```

When using the id-sha3-224, id-sha3-s256, id-sha3-384, or id-sha3-512 algorithm identifiers, the parameters field **MUST** be absent, not NULL but absent.

3. Signature Algorithms

This section specifies the conventions employed by CMS implementations that support the four SHA3 one-way hash functions with the RSASSA PKCS#1 version 1.5 signature algorithm [RFC8017] and the ECDSA [DSS] with the CMS signed-data content type.

Signature algorithm identifiers are located in the SignerInfo signatureAlgorithm field of SignedData. Also, signature algorithm identifiers are located in the SignerInfo signatureAlgorithm field of countersignature attributes.

Signature values are located in the SignerInfo signature field of SignedData. Also, signature values are located in the SignerInfo signature field of countersignature attributes.

3.1. RSASSA PKCS#1 v1.5 with SHA3

The RSASSA PKCS#1 v1.5 is defined in [RFC8017]. When RSASSA PKCS#1 v1.5 is used in conjunction with one of the SHA3 one-way hash functions, the object identifiers are:

```
OID ::= OBJECT IDENTIFIER

sigAlgs OBJECT IDENTIFIER ::= { joint-iso-itu-t(2) country(16)
    us(840) organization(1) gov(101) csor(3) nistAlgorithm(4) 3 }

id-rsassa-pkcs1-v1-5-with-sha3-224 OID ::= { sigAlgs 13 }

id-rsassa-pkcs1-v1-5-with-sha3-256 OID ::= { sigAlgs 14 }

id-rsassa-pkcs1-v1-5-with-sha3-384 OID ::= { sigAlgs 15 }

id-rsassa-pkcs1-v1-5-with-sha3-512 OID ::= { sigAlgs 16 }
```

The algorithm identifier for RSASSA PKCS#1 v1.5 subject public keys in certificates is specified in [RFC3279], and it is repeated here for convenience:

```
rsaEncryption OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 1 }
```

When the rsaEncryption, id-rsassa-pkcs1-v1-5-with-sha3-224, id-rsassa-pkcs1-v1-5-with-sha3-256, id-rsassa-pkcs1-v1-5-with-sha3-384, and id-rsassa-pkcs1-v1-5-with-sha3-512 algorithm identifiers are used, the AlgorithmIdentifier parameters field **MUST** contain NULL.

When the rsaEncryption algorithm identifier is used, the RSA public key, which is composed of a modulus and a public exponent, **MUST** be encoded using the RSAPublicKey type as specified in [RFC3279]. The output of this encoding is carried in the certificate subject public key. The definition of RSAPublicKey is repeated here for convenience:

```
RSAPublicKey ::= SEQUENCE {
   modulus INTEGER, -- n
   publicExponent INTEGER } -- e
```

When signing, the RSASSA PKCS#1 v1.5 signature algorithm generates a single value. That value is used directly as the signature value.

3.2. ECDSA with SHA3

The ECDSA is defined in [DSS]. When the ECDSA is used in conjunction with one of the SHA3 one-way hash functions, the object identifiers are:

```
sigAlgs OBJECT IDENTIFIER ::= { joint-iso-itu-t(2) country(16)
    us(840) organization(1) gov(101) csor(3) nistAlgorithm(4) 3 }

id-ecdsa-with-sha3-224 OBJECT IDENTIFIER ::= { sigAlgs 9 }

id-ecdsa-with-sha3-256 OBJECT IDENTIFIER ::= { sigAlgs 10 }

id-ecdsa-with-sha3-384 OBJECT IDENTIFIER ::= { sigAlgs 11 }

id-ecdsa-with-sha3-512 OBJECT IDENTIFIER ::= { sigAlgs 12 }
```

When using the id-ecdsa-with-sha3-224, id-ecdsa-with-sha3-256, id-ecdsa-with-sha3-384, and id-ecdsa-with-sha3-512 algorithm identifiers, the parameters field **MUST** be absent, not NULL but absent.

The conventions for ECDSA public keys are as specified in [RFC5480]. The ECParameters associated with the ECDSA public key in the signers certificate **SHALL** apply to the verification of the signature.

When signing, the ECDSA algorithm generates two values. These values are commonly referred to as r and s. To easily transfer these two values as one signature, they **MUST** be ASN.1 encoded using the ECDSA-Sig-Value defined in [RFC3279], which is repeated here for convenience:

```
ECDSA-Sig-Value ::= SEQUENCE {
    r  INTEGER,
    s  INTEGER }
```

4. Message Authentication Codes Using HMAC and SHA3

This section specifies the conventions employed by CMS implementations that support the Hashed Message Authentication Code (HMAC) [RFC2104] with SHA3 message authentication code (MAC).

MAC algorithm identifiers are located in the AuthenticatedData macAlgorithm field.

MAC values are located in the AuthenticatedData mac field.

When HMAC is used in conjunction with one of the SHA3 one-way hash functions, the object identifiers are:

```
hashAlgs OBJECT IDENTIFIER ::= { joint-iso-itu-t(2) country(16)
    us(840) organization(1) gov(101) csor(3) nistAlgorithm(4) 2 }

id-hmacWithSHA3-224 OBJECT IDENTIFIER ::= { hashAlgs 13 }

id-hmacWithSHA3-256 OBJECT IDENTIFIER ::= { hashAlgs 14 }

id-hmacWithSHA3-384 OBJECT IDENTIFIER ::= { hashAlgs 15 }

id-hmacWithSHA3-512 OBJECT IDENTIFIER ::= { hashAlgs 16 }
```

When the id-hmacWithSHA3-224, id-hmacWithSHA3-256, id-hmacWithSHA3-384, and id-hmacWithSHA3-512 algorithm identifiers are used, the parameters field **MUST** be absent, not NULL but absent.

5. Key Derivation Functions

The CMS KEMRecipientInfo structure [RFC9629] is one place where algorithm identifiers for key-derivation functions are needed.

5.1. HKDF with SHA3

This section assigns four algorithm identifiers that can be employed by CMS implementations that support the HMAC-based Extract-and-Expand Key Derivation Function (HKDF) [RFC5869] with the SHA3 family of hash functions.

When HKDF is used in conjunction with one of the SHA3 one-way hash functions, the object identifiers are:

```
id-alg OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) 3 }

id-alg-hkdf-with-sha3-224 OBJECT IDENTIFIER ::= { id-alg 32 }

id-alg-hkdf-with-sha3-256 OBJECT IDENTIFIER ::= { id-alg 33 }

id-alg-hkdf-with-sha3-384 OBJECT IDENTIFIER ::= { id-alg 34 }

id-alg-hkdf-with-sha3-512 OBJECT IDENTIFIER ::= { id-alg 35 }
```

When id-alg-hkdf-with-sha3-224, id-alg-hkdf-with-sha3-256, id-alg-hkdf-with-sha3-384, or id-alg-hkdf-with-sha3-512 is used in an algorithm identifier, the parameters field **MUST** be absent, not NULL but absent.

5.2. KMAC128-KDF and KMAC256-KDF

This section specifies the conventions employed by CMS implementations that employ either KMAC128 or KMAC256 as KDFs as defined in Section 4.4 of [NIST.SP.800-108r1-upd1].

KMAC128 and KMAC256 are specified in [NIST.SP.800-185]. The use of KMAC128 and KMAC256 as KDFs are defined as follows:

> KMAC128-KDF is KMAC128(K, X, L, S).
>
> KMAC256-KDF is KMAC256(K, X, L, S).

The parameters to the KMAC128 and KMAC256 functions are:

K The input key-derivation key. The length of K **MUST** be less than 2^{2040}.

X The context, which contains the ASN.1 DER encoding of CMSORIforKEMOtherInfo when the KDF is used with [RFC9629].

L The output length in bits. L **MUST** be greater than or equal to 0 and **MUST** be less than 2^{2040}.

S The optional customization label, such as "KDF" (0x4B4446). The length of S **MUST** be less than 2^{2040}.

The K parameter is known to all authorized parties; it is often the output of a KEM Decap() operation. The X parameter is assembled from data that is transmitted by the originator. The L parameter is determined by the size of the output keying material. The S parameter is optional, and if it is provided by the originator, it is passed in the parameters field of the KDF algorithm identifier.

When KMAC128-KDF or KMAC256-KDF is used, the object identifiers are:

```
hashAlgs OBJECT IDENTIFIER ::= { joint-iso-itu-t(2) country(16)
    us(840) organization(1) gov(101) csor(3) nistAlgorithm(4) 2 }

id-kmac128 OBJECT IDENTIFIER ::= { hashAlgs 21 }

id-kmac256 OBJECT IDENTIFIER ::= { hashAlgs 22 }
```

When id-kmac128 or id-kmac256 is used as part of an algorithm identifier, the parameters field **MUST** be absent when there is no customization label (S). If any value is provided for S, then the parameters field **MUST** be present and contain the value of S, encoded as Customization.

```
Customization ::= OCTET STRING
```

5.3. KDF2 and KDF3 with SHA3

This section specifies the conventions employed by CMS implementations that employ either the KDF2 or KDF3 functions defined in [ANS-X9.44]. The CMS KEMRecipientInfo structure [RFC9629] is one place where algorithm identifiers for key-derivation functions are needed.

The key-derivation function algorithm identifier is an object identifier and optional parameter. When KDF2 and KDF3 are used, they are identified by the id-kdf-kdf2 and id-kdf-kdf3 object identifiers, respectively. The key-derivation function algorithm identifier parameters carry a message digest algorithm identifier, which indicates the hash function that is being employed. To support SHA3, the key-derivation function algorithm identifier parameters contain an algorithm identifier from Section 2.

```
x9-44 OBJECT IDENTIFIER ::= { iso(1) identified-organization(3)
    tc68(133) country(16) x9(840) x9Standards(9) x9-44(44) }

x9-44-components OBJECT IDENTIFIER ::= { x9-44 components(1) }

id-kdf-kdf2 OBJECT IDENTIFIER ::= { x9-44-components kdf2(1) }

id-kdf-kdf3 OBJECT IDENTIFIER ::= { x9-44-components kdf3(2) }
```

6. Security Considerations

Implementations must protect the signer's private key. Compromise of the signer's private key permits masquerade.

Implementations must protect the key-derivation key. Compromise of the key-derivation key permits others to derive the derived keying material, which would result in loss of confidentiality, integrity, or authentication, depending on the use of the derived keying material.

When more than two parties share the same message-authentication key, data origin authentication is not assured. Any party that knows the message-authentication key can compute a valid MAC; therefore, the content could originate from any one of the parties.

Implementations must randomly generate message-authentication keys and one-time values, such as the k value when generating an ECDSA signature. In addition, the generation of public/private key pairs relies on a random numbers. The use of inadequate pseudorandom number generators (PRNGs) to generate cryptographic values can result in little or no security. Instead of brute-force searching the whole key space, an attacker may find it much easier to reproduce the PRNG environment that produced the keys and then search the resulting small set of possibilities. The generation of quality random numbers is difficult. [RFC4086] offers important guidance in this area, and Appendix 3 of FIPS PUB 186-4 [DSS] provides some PRNG techniques.

Implementers should be aware that cryptographic algorithms become weaker with time. As new cryptanalysis techniques are developed and computing performance improves, the work factor to break a particular cryptographic algorithm will reduce. Therefore, cryptographic algorithm implementations should be modular, allowing new algorithms to be readily inserted. That is, implementers should be prepared to regularly update the set of algorithms in their implementations.

7. IANA Considerations

IANA has assigned one object identifier for the ASN.1 module in Appendix A in the "SMI Security for S/MIME Module Identifiers (1.2.840.113549.1.9.16.0)" registry [IANA-MOD]:

```
id-mod-sha3-oids-2023 OBJECT IDENTIFIER ::= {
   iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
   pkcs-9(9) smime(16) mod(0) 78 }
```

IANA has assigned four object identifiers for the HKDF using SHA3 algorithm identifiers in the "SMI Security for S/MIME Algorithms (1.2.840.113549.1.9.16.3)" registry [IANA-ALG]:

```
id-alg-hkdf-with-sha3-224 OBJECT IDENTIFIER ::= { id-alg 32 }

id-alg-hkdf-with-sha3-256 OBJECT IDENTIFIER ::= { id-alg 33 }

id-alg-hkdf-with-sha3-384 OBJECT IDENTIFIER ::= { id-alg 34 }

id-alg-hkdf-with-sha3-512 OBJECT IDENTIFIER ::= { id-alg 35 }
```

8. References

8.1. Normative References

[ANS-X9.44] American National Standards Institute, "Public Key Cryptography for the Financial Services Industry -- Key Establishment Using Integer Factorization Cryptography", ANSI X9.44-2007 (R2017), 2017, <https://webstore.ansi.org/standards/ascx9/ansix9442007r2017>.

[DSS] National Institute of Standards and Technology, "Digital Signature Standard (DSS)", FIPS PUB 186-5, DOI 10.6028/NIST.FIPS.186-5, 3 February 2023, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf>.

[NIST.SP.800-108r1-upd1] Chen, L., "Recommendation for Key Derivation Using Pseudorandom Functions", NIST SP 800-108r1-upd1, DOI 10.6028/NIST.SP.800-108r1-upd1, 2 February 2024, <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-108r1-upd1.pdf>.

[NIST.SP.800-185] Kelsey, J., Chang, S., and R. Perlner, "SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash and ParallelHash", NIST SP 800-185, DOI 10.6028/NIST.SP.800-185, December 2016, <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-185.pdf>.

[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>.

[RFC3279] Bassham, L., Polk, W., and R. Housley, "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3279, DOI 10.17487/RFC3279, April 2002, <https://www.rfc-editor.org/info/rfc3279>.

[RFC5480] Turner, S., Brown, D., Yiu, K., Housley, R., and T. Polk, "Elliptic Curve Cryptography Subject Public Key Information", RFC 5480, DOI 10.17487/RFC5480, March 2009, <https://www.rfc-editor.org/info/rfc5480>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/info/rfc5652>.

[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>.

[RFC5912] Hoffman, P. and J. Schaad, "New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)", RFC 5912, DOI 10.17487/RFC5912, June 2010, <https://www.rfc-editor.org/info/rfc5912>.

[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>.

[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>.

[SHA3] National Institute of Standards and Technology, "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions", NIST FIPS 202, DOI 10.6028/NIST.FIPS.202, August 2015, <http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf>.

[X.680] ITU-T, "Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation", ITU-T Recommendation X.680, ISO/IEC 8824-1:2021, February 2021, <https://www.itu.int/rec/T-REC-X.680-202102-I/en>.

[X.690] ITU-T, "Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, ISO/IEC 8825-1:2021, February 2021, <https://www.itu.int/rec/T-REC-X.690-202102-I/en>.

8.2. Informative References

[IANA-ALG] IANA, "SMI Security for S/MIME Algorithms (1.2.840.113549.1.9.16.3)", <https://www.iana.org/assignments/smi-numbers/>.

[IANA-MOD] IANA, "SMI Security for S/MIME Module Identifier (1.2.840.113549.1.9.16.0)", <https://www.iana.org/assignments/smi-numbers/>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[RFC8702] Kampanakis, P. and Q. Dang, "Use of the SHAKE One-Way Hash Functions in the Cryptographic Message Syntax (CMS)", RFC 8702, DOI 10.17487/RFC8702, January 2020, <https://www.rfc-editor.org/info/rfc8702>.

[RFC9629] Housley, R., Gray, J., and T. Okubo, "Using Key Encapsulation Mechanism (KEM) Algorithms in the Cryptographic Message Syntax (CMS)", RFC 9629, DOI 10.17487/RFC9629, August 2024, <https://www.rfc-editor.org/info/rfc9629>.

Appendix A. ASN.1 Module

This section contains the ASN.1 module for the algorithm identifiers using the SHA3 family of hash functions [SHA3]. This module imports types from other ASN.1 modules that are defined in [RFC5912].

```
<CODE BEGINS>
   SHA3-OIDs-2023
     { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
       smime(16) modules(0) id-mod-sha3-oids-2023(78) }

   DEFINITIONS IMPLICIT TAGS ::=
   BEGIN

   EXPORTS ALL;

   IMPORTS

     AlgorithmIdentifier{}, DIGEST-ALGORITHM, SIGNATURE-ALGORITHM,
     KEY-DERIVATION, MAC-ALGORITHM
     FROM AlgorithmInformation-2009  -- [RFC5912]
       { iso(1) identified-organization(3) dod(6) internet(1)
         security(5) mechanisms(5) pkix(7) id-mod(0)
         id-mod-algorithmInformation-02(58) }

    mda-sha1, pk-rsa, pk-ec, ECDSA-Sig-Value
```

```
     FROM PKIXAlgs-2009  -- [RFC5912]
       { iso(1) identified-organization(3) dod(6) internet(1)
         security(5) mechanisms(5) pkix(7) id-mod(0)
         id-mod-pkix1-algorithms2008-02(56) }

     mda-sha224, mda-sha256, mda-sha384, mda-sha512
     FROM PKIX1-PSS-OAEP-Algorithms-2009  -- [RFC5912]
       { iso(1) identified-organization(3) dod(6) internet(1)
         security(5) mechanisms(5) pkix(7) id-mod(0)
         id-mod-pkix1-rsa-pkalgs-02(54) } ;


    --
    -- Alias
    --

    OID ::= OBJECT IDENTIFIER


    --
    -- Object Identifier Arcs
    --

    nistAlgorithm OID ::= { joint-iso-itu-t(2) country(16)
        us(840) organization(1) gov(101) csor(3) 4 }

    hashAlgs OID ::= { nistAlgorithm 2 }

    sigAlgs OID ::= { nistAlgorithm 3 }

    x9-44 OID ::= { iso(1) identified-organization(3) tc68(133)
        country(16) x9(840) x9Standards(9) x9-44(44) }

    x9-44-components OID ::= { x9-44 components(1) }

    id-alg OID ::= { iso(1) member-body(2) us(840)
        rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) 3 }


    --
    -- Message Digest Algorithms
    --

    id-sha3-224 OID ::= { hashAlgs 7 }

    id-sha3-256 OID ::= { hashAlgs 8 }

    id-sha3-384 OID ::= { hashAlgs 9 }

    id-sha3-512 OID ::= { hashAlgs 10 }

    mda-sha3-224 DIGEST-ALGORITHM ::= {
        IDENTIFIER id-sha3-224
        PARAMS ARE absent }

    mda-sha3-256 DIGEST-ALGORITHM ::= {
        IDENTIFIER id-sha3-256
        PARAMS ARE absent }
```

```
    mda-sha3-384 DIGEST-ALGORITHM ::= {
        IDENTIFIER id-sha3-384
        PARAMS ARE absent }

    mda-sha3-512 DIGEST-ALGORITHM ::= {
        IDENTIFIER id-sha3-512
        PARAMS ARE absent }

    HashAlgorithm ::= AlgorithmIdentifier{ DIGEST-ALGORITHM,
                          { HashAlgorithms } }

    HashAlgorithms DIGEST-ALGORITHM ::=  {
        mda-sha3-224 |
        mda-sha3-256 |
        mda-sha3-384 |
        mda-sha3-512,
        ... }


    --
    -- Signature Algorithms
    --

    id-rsassa-pkcs1-v1-5-with-sha3-224 OID ::= { sigAlgs 13 }

    id-rsassa-pkcs1-v1-5-with-sha3-256 OID ::= { sigAlgs 14 }

    id-rsassa-pkcs1-v1-5-with-sha3-384 OID ::= { sigAlgs 15 }

    id-rsassa-pkcs1-v1-5-with-sha3-512 OID ::= { sigAlgs 16 }

    id-ecdsa-with-sha3-224 OID ::= { sigAlgs 9 }

    id-ecdsa-with-sha3-256 OID ::= { sigAlgs 10 }

    id-ecdsa-with-sha3-384 OID ::= { sigAlgs 11 }

    id-ecdsa-with-sha3-512 OID ::= { sigAlgs 12 }

    sa-rsaWithSHA3-224 SIGNATURE-ALGORITHM ::= {
        IDENTIFIER id-rsassa-pkcs1-v1-5-with-sha3-224
        PARAMS TYPE NULL ARE required
        HASHES { mda-sha3-224 }
        PUBLIC-KEYS { pk-rsa }
        SMIME-CAPS {IDENTIFIED BY
            id-rsassa-pkcs1-v1-5-with-sha3-224 } }

    sa-rsaWithSHA3-256 SIGNATURE-ALGORITHM ::= {
        IDENTIFIER id-rsassa-pkcs1-v1-5-with-sha3-256
        PARAMS TYPE NULL ARE required
        HASHES { mda-sha3-256 }
        PUBLIC-KEYS { pk-rsa }
        SMIME-CAPS {IDENTIFIED BY
            id-rsassa-pkcs1-v1-5-with-sha3-256 } }

    sa-rsaWithSHA3-384 SIGNATURE-ALGORITHM ::= {
        IDENTIFIER id-rsassa-pkcs1-v1-5-with-sha3-384
```

```
        PARAMS TYPE NULL ARE required
        HASHES { mda-sha3-384 }
        PUBLIC-KEYS { pk-rsa }
        SMIME-CAPS {IDENTIFIED BY
            id-rsassa-pkcs1-v1-5-with-sha3-384 } }

    sa-rsaWithSHA3-512 SIGNATURE-ALGORITHM ::= {
        IDENTIFIER id-rsassa-pkcs1-v1-5-with-sha3-512
        PARAMS TYPE NULL ARE required
        HASHES { mda-sha3-512 }
        PUBLIC-KEYS { pk-rsa }
        SMIME-CAPS {IDENTIFIED BY
            id-rsassa-pkcs1-v1-5-with-sha3-512 } }

    sa-ecdsaWithSHA3-224 SIGNATURE-ALGORITHM ::= {
        IDENTIFIER id-ecdsa-with-sha3-224
        VALUE ECDSA-Sig-Value
        PARAMS ARE absent
        HASHES { mda-sha3-224 }
        PUBLIC-KEYS { pk-ec }
        SMIME-CAPS {IDENTIFIED BY id-ecdsa-with-sha3-224 } }

    sa-ecdsaWithSHA3-256 SIGNATURE-ALGORITHM ::= {
        IDENTIFIER id-ecdsa-with-sha3-256
        VALUE ECDSA-Sig-Value
        PARAMS ARE absent
        HASHES { mda-sha3-256 }
        PUBLIC-KEYS { pk-ec }
        SMIME-CAPS {IDENTIFIED BY id-ecdsa-with-sha3-256 } }

    sa-ecdsaWithSHA3-384 SIGNATURE-ALGORITHM ::= {
        IDENTIFIER id-ecdsa-with-sha3-384
        VALUE ECDSA-Sig-Value
        PARAMS ARE absent
        HASHES { mda-sha3-384 }
        PUBLIC-KEYS { pk-ec }
        SMIME-CAPS {IDENTIFIED BY id-ecdsa-with-sha3-384 } }

    sa-ecdsaWithSHA3-512 SIGNATURE-ALGORITHM ::= {
        IDENTIFIER id-ecdsa-with-sha3-512
        VALUE ECDSA-Sig-Value
        PARAMS ARE absent
        HASHES { mda-sha3-512 }
        PUBLIC-KEYS { pk-ec }
        SMIME-CAPS {IDENTIFIED BY id-ecdsa-with-sha3-512 } }

    SignatureAlg ::= AlgorithmIdentifier{ SIGNATURE-ALGORITHM,
                          { SignatureAlgs } }

    SignatureAlgs SIGNATURE-ALGORITHM ::= {
        sa-rsaWithSHA3-224 |
        sa-rsaWithSHA3-256 |
        sa-rsaWithSHA3-384 |
        sa-rsaWithSHA3-512 |
        sa-ecdsaWithSHA3-224 |
        sa-ecdsaWithSHA3-256 |
        sa-ecdsaWithSHA3-384 |
        sa-ecdsaWithSHA3-512,
```

```
        ... }


    --
    -- Message Authentication Codes
    --

    id-hmacWithSHA3-224 OID ::= { hashAlgs 13 }

    id-hmacWithSHA3-256 OID ::= { hashAlgs 14 }

    id-hmacWithSHA3-384 OID ::= { hashAlgs 15 }

    id-hmacWithSHA3-512 OID ::= { hashAlgs 16 }

    maca-hmacWithSHA3-224 MAC-ALGORITHM ::= {
        IDENTIFIER id-hmacWithSHA3-224
        PARAMS ARE absent
        IS-KEYED-MAC TRUE
        SMIME-CAPS {IDENTIFIED BY id-hmacWithSHA3-224 } }

    maca-hmacWithSHA3-256 MAC-ALGORITHM ::= {
        IDENTIFIER id-hmacWithSHA3-256
        PARAMS ARE absent
        IS-KEYED-MAC TRUE
        SMIME-CAPS {IDENTIFIED BY id-hmacWithSHA3-256 } }

    maca-hmacWithSHA3-384 MAC-ALGORITHM ::= {
        IDENTIFIER id-hmacWithSHA3-384
        PARAMS ARE absent
        IS-KEYED-MAC TRUE
        SMIME-CAPS {IDENTIFIED BY id-hmacWithSHA3-384 } }

    maca-hmacWithSHA3-512 MAC-ALGORITHM ::= {
        IDENTIFIER id-hmacWithSHA3-512
        PARAMS ARE absent
        IS-KEYED-MAC TRUE
        SMIME-CAPS {IDENTIFIED BY id-hmacWithSHA3-512 } }

    MACAlgorithm ::= AlgorithmIdentifier{ MAC-ALGORITHM,
                        { MACAlgorithms } }

    MACAlgorithms MAC-ALGORITHM ::= {
        maca-hmacWithSHA3-224 |
        maca-hmacWithSHA3-256 |
        maca-hmacWithSHA3-384 |
        maca-hmacWithSHA3-512,
        ... }


    --
    -- Key Derivation Algorithms
    --

    id-alg-hkdf-with-sha3-224 OID ::= { id-alg 32 }

    id-alg-hkdf-with-sha3-256 OID ::= { id-alg 33 }
```

```
    id-alg-hkdf-with-sha3-384 OID ::= { id-alg 34 }

    id-alg-hkdf-with-sha3-512 OID ::= { id-alg 35 }

    id-kmac128 OID ::= { hashAlgs 21 }

    id-kmac256  OID ::= { hashAlgs 22 }

    id-kdf-kdf2 OID ::= { x9-44-components kdf2(1) }

    id-kdf-kdf3 OID ::= { x9-44-components kdf3(2) }

    kda-hkdf-with-sha3-224 KEY-DERIVATION ::= {
        IDENTIFIER id-alg-hkdf-with-sha3-224
        PARAMS ARE absent
        -- No S/MIME caps defined -- }

    kda-hkdf-with-sha3-256 KEY-DERIVATION ::= {
        IDENTIFIER id-alg-hkdf-with-sha3-256
        PARAMS ARE absent
        -- No S/MIME caps defined -- }

    kda-hkdf-with-sha3-384 KEY-DERIVATION ::= {
        IDENTIFIER id-alg-hkdf-with-sha3-384
        PARAMS ARE absent
        -- No S/MIME caps defined -- }

    kda-hkdf-with-sha3-512 KEY-DERIVATION ::= {
        IDENTIFIER id-alg-hkdf-with-sha3-512
        PARAMS ARE absent
        -- No S/MIME caps defined -- }

    kda-kmac128 KEY-DERIVATION ::= {
        IDENTIFIER id-kmac128
        PARAMS TYPE Customization ARE optional
        -- PARAMS are absent when Customization is ''H --
        -- No S/MIME caps defined -- }

    kda-kmac256 KEY-DERIVATION ::= {
        IDENTIFIER id-kmac256
        PARAMS TYPE Customization ARE optional
        -- PARAMS are absent when Customization is ''H --
        -- No S/MIME caps defined -- }

    kda-kdf2 KEY-DERIVATION ::= {
        IDENTIFIER id-kdf-kdf2
        PARAMS TYPE KDF2-HashFunction ARE required
        -- No S/MIME caps defined -- }

    kda-kdf3 KEY-DERIVATION ::= {
        IDENTIFIER id-kdf-kdf3
        PARAMS TYPE KDF3-HashFunction ARE required
        -- No S/MIME caps defined -- }

    Customization ::= OCTET STRING

    KDF2-HashFunction ::= AlgorithmIdentifier { DIGEST-ALGORITHM,
                              { KDF2-HashFunctions } }
```

```
   KDF2-HashFunctions DIGEST-ALGORITHM ::= {
      X9-HashFunctions,
      ... }

   KDF3-HashFunction ::= AlgorithmIdentifier { DIGEST-ALGORITHM,
                             { KDF3-HashFunctions } }

   KDF3-HashFunctions DIGEST-ALGORITHM ::= {
      X9-HashFunctions,
      ... }

   X9-HashFunctions DIGEST-ALGORITHM ::= {
       mda-sha1 |
       mda-sha224 |
       mda-sha256 |
       mda-sha384 |
       mda-sha512 |
       mda-sha3-224 |
       mda-sha3-256 |
       mda-sha3-384 |
       mda-sha3-512,
       ... }

   KeyDerivationFunction ::=  AlgorithmIdentifier{ KEY-DERIVATION,
                                  { KeyDevAlgs } }

   KeyDevAlgs KEY-DERIVATION ::= {
       kda-hkdf-with-sha3-224 |
       kda-hkdf-with-sha3-256 |
       kda-hkdf-with-sha3-384 |
       kda-hkdf-with-sha3-512 |
       kda-kmac128 |
       kda-kmac256 |
       kda-kdf2 |
       kda-kdf3,
       ... }

   END

<CODE ENDS>
```

Acknowledgements

Thanks to Daniel Van Geest and Sean Turner for their careful review and thoughtful comments.

Thanks to Sara Kerman, Quynh Dang, and David Cooper for getting the object identifiers assigned for KMAC128 and KMAC256.

Author's Address

Russ Housley
Vigil Security, LLC
Herndon, VA
United States of America
Email: housley@vigilsec.com