LAMPS WG
Internet-Draft
Intended status: Standards Track
Expires: May 29, 2019
Q. Dang
NIST
P. Kampanakis
Cisco Systems
November 25, 2018

Use of the SHAKE One-way Hash Functions in the Cryptographic Message Syntax (CMS)
draft-ietf-lamps-cms-shakes-03

Abstract

This document describes the conventions for using the SHAKE family of hash functions with the Cryptographic Message Syntax (CMS) as one-way hash functions with the RSA Probabilistic signature and ECDSA signature algorithms, as message digests and message authentication codes. The conventions for the associated signer public keys in CMS are also described.

Status of This Memo

Copyright Notice

Table of Contents

1. Change Log

[EDNOTE: Remove this section before publication.]

- draft-ietf-lamps-cms-shake-03:

  * Removed paragraph suggesting KMAC to be used in generating k in Deterministic ECDSA. That should be RFC6979-bis.

  * Removed paragraph from Security Considerations that talks about randomness of k because we are using deterministic ECDSA.

  * Completed ASN.1 module and fixed KMAC ASN.1 based on Jim's feedback.

  * Text fixes.

- draft-ietf-lamps-cms-shake-02:

  * Updates based on suggestions and clarifications by Jim.

* Started ASN.1 module.

o draft-ietf-lamps-cms-shake-01:

* Significant reorganization of the sections to simplify the introduction, the new OIDs and their use in CMS.

* Added new OIDs for RSASSA-PSS that hardcodes hash, salt and MGF, according the WG consensus.

* Updated Public Key section to use the new RSASSA-PSS OIDs and clarify the algorithm identifier usage.

* Removed the no longer used SHAKE OIDs from section 3.1.

o draft-ietf-lamps-cms-shake-00:

* Various updates to title and section names.

* Content changes filling in text and references.

o draft-dang-lamps-cms-shakes-hash-00:

* Initial version

2. 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 SHAKE128 and SHAKE256 specified in [SHA3] as new hash functions in CMS. In addition, it describes the use of these functions with the RSASSA-PSS signature algorithm [RFC8017] and the Elliptic Curve Digital Signature Algorithm (ECDSA) [X9.62] with the CMS signed-data content type.

In the SHA-3 family, two extendable-output functions (SHAKEs), SHAKE128 and SHAKE256, are defined. Four other hash function instances, SHA3-224, SHA3-256, SHA3-384, and SHA3-512 are also defined but are out of scope for this document. A SHAKE is a variable length hash function. The output length, in bits, of a SHAKE is defined by the d parameter. The corresponding collision and second preimage resistance strengths for SHAKE128 are min(d/2,128) and min(d,128) bits respectively. And, the corresponding collision and second preimage resistance strengths for SHAKE256 are min(d/2,256) and min(d,256) bits respectively.

A SHAKE can be used in CMS as the message digest function (to hash the message to be signed) in RSASSA-PSS and deterministic ECDSA,

message authentication code and as the mask generating function in RSASSA-PSS. This specification describes the identifiers for SHAKEs to be used in CMS and their meaning.

2.1. Terminology

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

3. Identifiers

This section defines six new OIDs for using SHAKE128 and SHAKE256 in CMS.

EDNOTE: If PKIX draft is standardized first maybe we should not say the identifiers are new for the RSASSA-PSS and ECDSA.

Two object identifiers for SHAKE128 and SHAKE256 hash functions are defined in [shake-nist-oids] and we include them here for convenience.

```
id-shake128-len OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
     country(16) us(840) organization(1) gov(101) csor(3)
     nistAlgorithm(4) 2 17 }

id-shake256-len OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
     country(16) us(840) organization(1) gov(101) csor(3)
     nistAlgorithm(4) 2 18 }
```

In this specification, when using the id-shake128-len or id-shake256-len algorithm identifiers, the parameters MUST be absent. That is, the identifier SHALL be a SEQUENCE of one component, the OID.

We define two new identifiers for RSASSA-PSS signatures using SHAKEs.

```
id-RSASSA-PSS-SHAKE128  OBJECT IDENTIFIER  ::=  { TBD }

id-RSASSA-PSS-SHAKE256  OBJECT IDENTIFIER  ::=  { TBD }

[ EDNOTE: "TBD" will be specified by NIST later. ]
```

The same RSASSA-PSS algorithm identifiers can be used for identifying public keys and signatures.

We define two new algorithm identifiers of ECDSA signatures using SHAKEs.

```
id-ecdsa-with-SHAKE128 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
    country(16) us(840) organization(1) gov(101) csor(3)
    nistAlgorithm(4) 3  TBD }

id-ecdsa-with-SHAKE256 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
    country(16) us(840) organization(1) gov(101) csor(3)
    nistAlgorithm(4) 3  TBD }
```

[EDNOTE: "TBD" will be specified by NIST.]

The parameters for the four RSASSA-PSS and deterministic ECDSA identifiers MUST be absent. That is, each identifier SHALL be a SEQUENCE of one component, the OID.

Two new object identifiers for KMACs using SHAKE128 and SHAKE256 are define elow.

```
id-KmacWithSHAKE128 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
    country(16) us(840) organization(1) gov(101) csor(3)
    nistAlgorithm(4) 2 19 }

id-KmacWithSHAKE256 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
    country(16) us(840) organization(1) gov(101) csor(3)
    nistAlgorithm(4) 2 20 }
```

The parameters for id-KmacWithSHAKE128 and id-KmacWithSHAKE256 MUST be absent. That is, each identifier SHALL be a SEQUENCE of one component, the OID.

Section 4.1, Section 4.2.1, Section 4.2.2 and Section 4.4 specify the required output length for each use of SHAKE128 or SHAKE256 in message digests, RSASSA-PSS, determinstic ECDSA and KMAC.

4. Use in CMS

4.1. Message Digests

The id-shake128-len and id-shake256-len OIDs (Section 3) can be used as the digest algorithm identifiers located in the SignedData, SignerInfo, DigestedData, and the AuthenticatedData digestAlgorithm fields in CMS [RFC5652]. The encoding MUST omit the parameters field and the output size, d, for the SHAKE128 or SHAKE256 message digest MUST be 256 or 512 bits respectively.

The digest values are located in the DigestedData field and the Message Digest authenticated attribute included in the signedAttributes of the SignedData signerInfo. In addition, digest

values are input to signature algorithms. The digest algorithm MUST be the same as the message hash algorithms used in signatures.

4.2. Signatures

In CMS, signature algorithm identifiers are located in the SignerInfo signatureAlgorithm field of SignedData content type and countersignature attribute. Signature values are located in the SignerInfo signature field of SignedData and countersignature.

Conforming implementations that process RSASSA-PSS and deterministic ECDSA with SHAKE signatures when processing CMS data MUST recognize the corresponding OIDs specified in Section 3.

4.2.1. RSASSA-PSS Signatures

The RSASSA-PSS algorithm is defined in [RFC8017]. When id-RSASSA-PSS-SHAKE128 or id-RSASSA-PSS-SHAKE256 specified in Section 3 is used, the encoding MUST omit the parameters field. That is, the AlgorithmIdentifier SHALL be a SEQUENCE of one component, id-RSASSA-PSS-SHAKE128 or id-RSASSA-PSS-SHAKE256.

The hash algorithm to hash a message being signed and the hash and the hash algorithm as the mask generation function used in RSASSA-PSS MUST be the same, SHAKE128 or SHAKE256 respectively. The output-length of the hash algorithm which hashes the message SHALL be 32 or 64 bytes respectively.

The mask generation function takes an octet string of variable length and a desired output length as input, and outputs an octet string of the desired length. In RSASSA-PSS with SHAKES, the SHAKEs MUST be used natively as the MGF function, instead of the MGF1 algorithm that uses the hash function in multiple iterations as specified in Section B.2.1 of [RFC8017]. In other words, the MGF is defined as the SHAKE128 or SHAKE256 output of the mgfSeed for id-RSASSA-PSS-SHAKE128 and id-RSASSA-PSS-SHAKE256 respectively. The mgfSeed is the seed from which mask is generated, an octet string [RFC8017]. The output length is (n - 264)/8 or (n - 520)/8 bytes respectively, where n is the RSA modulus in bits. For example, when RSA modulus n is 2048, the output length of SHAKE128 or SHAKE256 as the MGF will be 223 or 191-bits when id-RSASSA-PSS-SHAKE128 or id-RSASSA-PSS-SHAKE256 is used respectively.

The RSASSA-PSS saltLength MUST be 32 or 64 bytes respectively. Finally, the trailerField MUST be 1, which represents the trailer field with hexadecimal value 0xBC [RFC8017].

4.2.2. Deterministic ECDSA Signatures

The Elliptic Curve Digital Signature Algorithm (ECDSA) is defined in [X9.62]. When the id-ecdsa-with-SHAKE128 or id-ecdsa-with-SHAKE256 (specified in Section 3) algorithm identifier appears, the respective SHAKE function is used as the hash. The encoding MUST omit the parameters field. That is, the AlgorithmIdentifier SHALL be a SEQUENCE of one component, the OID id-ecdsa-with-SHAKE128 or id-ecdsa-with-SHAKE256.

For simplicity and compliance with the ECDSA standard specification, the output size of the hash function must be explicitly determined. The output size, d, for SHAKE128 or SHAKE256 used in ECDSA MUST be 256 or 512 bits respectively.

Conforming implementations that generate ECDSA with SHAKE signatures in CMS MUST generate such signatures with a deterministicly generated, non-random k in accordance with all the requirements specified in [RFC6979]. They MAY also generate such signatures in accordance with all other recommendations in [X9.62] or [SEC1] if they have a stated policy that requires conformance to these standards.

4.3. Public Keys

In CMS, the signer's public key algorithm identifiers are located in the OriginatorPublicKey's algorithm attribute.

Conforming implementations MUST specify the algorithms explicitly by using the OIDs specified in Section 3 when encoding RSASSA-PSS and ECDSA with SHAKE public keys in CMS messages. The conventions for RSASSA-PSS and ECDSA public keys algorithm identifiers are as specified in [RFC3279], [RFC4055] and [RFC5480] , but we include them below for convenience.

4.3.1. RSASSA-PSS Public Keys

[RFC3279] defines the following OID for RSA AlgorithmIdentifier in the SubjectPublicKeyInfo with NULL parameters.

```
rsaEncryption OBJECT IDENTIFIER ::=  { pkcs-1 1}
```

Additionally, when the RSA private key owner wishes to limit the use of the public key exclusively to RSASSA-PSS, the AlgorithmIdentifier for RSASSA-PSS defined in Section 3 can be used as the algorithm attribute in the OriginatorPublicKey sequence. The identifier parameters, as explained in Section 3, MUST be absent. The RSASSA-

PSS algorithm functions and output lengths are the same as defined in Section 4.2.1.

Regardless of what public key 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 [RFC4055]. The output of this encoding is carried in the CMS publicKey bit string.

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

4.3.2. ECDSA Public Keys

For ECDSA, the mandatory EC SubjectPublicKey is defined in Section 2.1.1 and its syntax in Section 2.2 of [RFC5480]. We also include them here for convenience:

```
id-ecPublicKey OBJECT IDENTIFIER ::= {
    iso(1) member-body(2) us(840) ansi-X9-62(10045) keyType(2) 1 }

ECParameters ::= CHOICE {
    namedCurve         OBJECT IDENTIFIER
    -- implicitCurve   NULL
    -- specifiedCurve  SpecifiedECDomain
    }
```

The ECParameters associated with the ECDSA public key in the signers certificate SHALL apply to the verification of the signature.

4.4. Message Authentication Codes

KMAC message authentication code (KMAC) is specified in [SP800-185]. In CMS, KMAC algorithm identifiers are located in the AuthenticatedData macAlgorithm field. The KMAC values are located in the AuthenticatedData mac field.

When the id-KmacWithSHAKE128 or id-KmacWithSHAKE256 algorithm identifier is used as the MAC algorithm identifier, the parameters field is optional (absent or present). If absent, the SHAKE256 output length used in KMAC is 256 or 512 bits respectively and the customization string is an empty string by default.

Conforming implementations that process KMACs with the SHAKEs when processing CMS data MUST recognize these identifiers.

When calculating the KMAC output, the variable N is 0xD2B282C2, S is an empty string, and L, the integer representing the requested output length in bits, is 256 or 512 for KmacWithSHAKE128 or KmacWithSHAKE256 respectively in this specification.

5. IANA Considerations

[EDNOTE: Update here only if there are OID allocations by IANA.]

This document has no IANA actions.

6. Security Considerations

The SHAKEs are deterministic functions. Like any other deterministic function, executing each function with the same input multiple times will produce the same output. Therefore, users should not expect unrelated outputs (with the same or different output lengths) from excuting a SHAKE function with the same input multiple times. The shorter one of any 2 outputs produced from a SHAKE with the same input is a prefix of the longer one. It is a similar situation as truncating a 512-bit output of SHA-512 by taking its 256 left-most bits. These 256 left-most bits are a prefix of the 512-bit output.

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

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

Implementers should be aware that cryptographic algorithms may become weaker with time. As new cryptanalysis techniques are developed and computing power increases, the work factor or time required to break a particular cryptographic algorithm may decrease. 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. Acknowledgements

This document is based on Russ Housley's draft [I-D.housley-lamps-cms-sha3-hash] It replaces SHA3 hash functions by SHAKE128 and SHAKE256 as the LAMPS WG agreed.

8. References

8.1. Normative References

[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>.

[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>.

[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>.

[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>.

[SHA3] National Institute of Standards and Technology, U.S. Department of Commerce, "SHA-3 Standard - Permutation-Based Hash and Extendable-Output Functions", FIPS PUB 202, August 2015.

[SP800-185]
National Institute of Standards and Technology, "SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash and ParallelHash. NIST SP 800-185", December 2016, <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-185.pdf>.

8.2. Informative References

[I-D.housley-lamps-cms-sha3-hash]
Housley, R., "Use of the SHA3 One-way Hash Functions in the Cryptographic Message Syntax (CMS)", draft-housley-lamps-cms-sha3-hash-00 (work in progress), March 2017.

[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>.

[RFC6979] Pornin, T., "Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)", RFC 6979, DOI 10.17487/RFC6979, August 2013, <https://www.rfc-editor.org/info/rfc6979>.

[SEC1] Standards for Efficient Cryptography Group, "SEC 1: Elliptic Curve Cryptography", May 2009, <http://www.secg.org/sec1-v2.pdf>.

[shake-nist-oids] National Institute of Standards and Technology, "Computer Security Objects Register", October 2017, <https://csrc.nist.gov/Projects/Computer-Security-Objects-Register/Algorithm-Registration>.

[X9.62] American National Standard for Financial Services (ANSI), "X9.62-2005 Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Standard (ECDSA)", November 2005.

Appendix A. ASN.1 Module

This appendix includes the ASN.1 modules for SHAKEs in CMS. This module includes some ASN.1 from other standards for reference.

```
CMSAlgsForSHAKE-2018 { { iso(1) member-body(2) us(840)
     rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) modules(0)
     id-mod-cms-shakes(TBD) }

DEFINITIONS EXPLICIT TAGS ::=

BEGIN

-- EXPORTS ALL;

IMPORTS

DIGEST-ALGORITHM, MAC-ALGORITHM, SMIME-CAPS
FROM AlgorithmInformation-2009
  { iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) id-mod(0)
    id-mod-algorithmInformation-02(58) }
```

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

   --
   -- Message Digest Algorithms (mda-)
   -- used in SignedData, SignerInfo, DigestedData,
   -- and the AuthenticatedData digestAlgorithm
   -- fields in CMS
   --
   digestAlgorithms DIGEST-ALGORITHM ::= {
    ...
    -- This expands MessageAuthAlgs from [RFC5652]
    -- and MessageDigestAlgs in [RFC5753]
    mda-shake128   |
    mda-shake256,
    ...
   }

   --
   -- One-Way Hash Functions
   -- SHAKE128
   mda-shake128 DIGEST-ALGORITHM ::= {
     IDENTIFIER id-shake128  -- with output length 32 bytes.
   }
   id-shake128 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2) country(16)
                                       us(840) organization(1) gov(101)
                                       csor(3) nistAlgorithm(4)
                                       hashAlgs(2) 11 }

   -- SHAKE-256
   mda-shake256 DIGEST-ALGORITHM ::= {
     IDENTIFIER id-shake256  -- with output length 64 bytes.
   }
   id-shake256 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2) country(16)
                                       us(840) organization(1) gov(101)
                                       csor(3) nistAlgorithm(4)
                                       hashAlgs(2) 12 }

   --
   -- Public key algorithm identifiers located in the
   -- OriginatorPublicKey's algorithm attribute in CMS.
   -- And Signature identifiers used in SignerInfo
   -- signatureAlgorithm field of SignedData content
   -- type and countersignature attribute in CMS.
   --
   -- From RFC5280, for reference.
```

```
   -- rsaEncryption OBJECT IDENTIFIER ::=  { pkcs-1 1 }
      -- When the rsaEncryption algorithm identifier is used
      -- for a public key, the AlgorithmIdentifier parameters
      -- field MUST contain NULL.
   --
   id-RSASSA-PSS-SHAKE128  OBJECT IDENTIFIER  ::=  { TBD }
   id-RSASSA-PSS-SHAKE256  OBJECT IDENTIFIER  ::=  { TBD }
      -- When the id-RSASSA-PSS-* algorithm identifiers are used
      -- for a public key or a signature in CMS, the AlgorithmIdentifier
      -- parameters field MUST be absent. The message digest algorithm
      -- used in RSASSA-PSS MUST be SHAKE128 or SHAKE256 with a 32 or
      -- 64 byte outout length respectively. The mask generating
      -- function MUST be SHAKE128 or SHAKE256 with an output length
      -- of (n - 264)/8 or (n - 520)/8 bytes respectively, where n
      -- is the RSA modulus in bits.  The RSASSA-PSS saltLength MUST
      -- be 32 or 64 bytes respectively. In both cases, the RSA
      -- public key, MUST be encoded using the RSAPublicKey type.
   -- From RFC4055, for reference.
   -- RSAPublicKey ::= SEQUENCE {
   --   modulus INTEGER, -- n
   --   publicExponent INTEGER } -- e

   id-ecdsa-with-shake128 ::= { joint-iso-itu-t(2) country(16)
                                us(840) organization(1) gov(101)
                                csor(3) nistAlgorithm(4)
                                sigAlgs(3) TBD }
   id-ecdsa-with-shake256 ::= { joint-iso-itu-t(2) country(16)
                                us(840) organization(1) gov(101)
                                csor(3) nistAlgorithm(4)
                                sigAlgs(3) TBD }
      -- When the id-ecdsa-with-shake* algorithm identifiers are
      -- used in CMS, the AlgorithmIdentifier parameters field
      -- MUST be absent and the signature algorithm should
      -- Deterministic ECDSA [RFC6979]. The message digest MUST
      -- be SHAKE128 or SHAKE256 with a 32 or 64 byte outout
      -- length respectively. In both cases, the ECDSA public key,
      -- MUST be encoded using the id-ecPublicKey type.
   -- From RFC5480, for reference.
   -- id-ecPublicKey OBJECT IDENTIFIER ::= {
   --     iso(1) member-body(2) us(840) ansi-X9-62(10045) keyType(2) 1 }
     -- The id-ecPublicKey parameters must be absent or present
     -- and are defined as
   -- ECParameters ::= CHOICE {
   --     namedCurve         OBJECT IDENTIFIER
   --     -- implicitCurve   NULL
   --     -- specifiedCurve  SpecifiedECDomain
   --  }
```

```
--
-- Message Authentication (maca-) Algorithms
-- used in AuthenticatedData macAlgorithm in CMS
--
MessageAuthAlgs MAC-ALGORITHM ::= {
  ...
      -- This expands MessageAuthAlgs from [RFC5652] and [RFC6268]
      maca-KMACwithSHAKE128   |
      maca-KMACwithSHAKE256
}

SMimeCaps SMIME-CAPS ::= {
   ...
       -- The expands SMimeCaps from [RFC5911]
   maca-KMACwithSHAKE128   |
   maca-KMACwithSHAKE256
 }

--
-- KMAC with SHAKE128
maca-KMACwithSHAKE128 MAC-ALGORITHM ::= {
      IDENTIFIER id-KMACWithSHAKE128
      PARAMS TYPE KMACwithSHAKE128-params ARE optional
        -- If KMACwithSHAKE128-params parameters are absent
        -- the SHAKE128 output length used in KMAC is 256 bits
        -- and the customization string is an empty string.
      SMIME-CAPS {IDENTIFIED BY id-KMACWithSHAKE128}
}
id-KMACWithSHAKE128 OBJECT IDENTIFIER ::=  { joint-iso-itu-t(2)
                             country(16) us(840) organization(1)
                             gov(101) csor(3) nistAlgorithm(4)
                             hashAlgs(2) 19 }
KMACwithSHAKE128-params ::= SEQUENCE {
   KMACOutputLength     INTEGER DEFAULT 256, -- Output length in bits
   customizationString  OCTET STRING DEFAULT ''H
}

-- KMAC with SHAKE256
maca-KMACwithSHAKE256 MAC-ALGORITHM ::= {
      IDENTIFIER id-KMACWithSHAKE256
      PARAMS TYPE KMACwithSHAKE256-params ARE optional
         -- If KMACwithSHAKE256-params parameters are absent
         -- the SHAKE256 output length used in KMAC is 512 bits
         -- and the customization string is an empty string.
      SMIME-CAPS {IDENTIFIED BY id-KMACWithSHAKE256}
}
id-KMACWithSHAKE256 OBJECT IDENTIFIER ::=  { joint-iso-itu-t(2)
                            country(16) us(840) organization(1)
```

```
                              gov(101) csor(3) nistAlgorithm(4)
                              hashAlgs(2) 20 }
   KMACwithSHAKE256-params ::= SEQUENCE {
      KMACOutputLength     INTEGER DEFAULT 512, -- Output length in bits
      customizationString  OCTET STRING DEFAULT ''H
   }

   END
```

Authors' Addresses

Quynh Dang
NIST
100 Bureau Drive
Gaithersburg, MD 20899

Email: quynh.Dang@nist.gov

Panos Kampanakis
Cisco Systems

Email: pkampana@cisco.com