Internet X.509 Public Key Infrastructure: Additional Algorithm Identifiers for RSASSA-PSS and ECDSA using SHAKEs
draft-ietf-lamps-pkix-shake-08

Abstract

   Digital signatures are used to sign messages, X.509 certificates and
   CRLs (Certificate Revocation Lists). This document describes the
   conventions for using the SHAKE function family in Internet X.509
   certificates and CRLs as one-way hash functions with the RSA
   Probabilistic signature and ECDSA signature algorithms. The
   conventions for the associated subject public keys are also
   described.

Status of This Memo

   This Internet-Draft is submitted in full conformance with the
   provisions of BCP 78 and BCP 79.

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF). Note that other groups may also distribute
   working documents as Internet-Drafts. The list of current Internet-
   Drafts is at https://datatracker.ietf.org/drafts/current/.

   Internet-Drafts are draft documents valid for a maximum of six months
   and may be updated, replaced, or obsoleted by other documents at any
   time. It is inappropriate to use Internet-Drafts as reference
   material or to cite them other than as "work in progress."

   This Internet-Draft will expire on August 4, 2019.

Copyright Notice

Table of Contents

1. Change Log

   [EDNOTE: Remove this section before publication.]

   o draft-ietf-lamps-pkix-shake-08:

      * Small nits from Russ while in WGLC.

   o draft-ietf-lamps-pkix-shake-07:

      * Incorporated Eric's suggestion from WGLC.

   o draft-ietf-lamps-pkix-shake-06:

      * Added informative references.

      * Updated ASN.1 so it compiles.

      * Updated IANA considerations.

   o draft-ietf-lamps-pkix-shake-05:

      * Added RFC8174 reference and text.

   * Explicitly explained why RSASSA-PSS-params are omitted in
      section 5.1.1.

   * Simplified Public Keys section by removing redundand info from
      RFCs.

o draft-ietf-lamps-pkix-shake-04:

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

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

   * Various ASN.1 fixes.

   * Text fixes.

o draft-ietf-lamps-pkix-shake-03:

   * Updates based on suggestions and clarifications by Jim.

   * Added ASN.1.

o draft-ietf-lamps-pkix-shake-02:

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

   * Added new OIDs for RSASSA-PSS that hardcode 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.

   * Consolidated subsection for message digest algorithms.

   * Text fixes.

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

   * Changed titles and section names.

   * Removed DSA after WG discussions.

   * Updated shake OID names and parameters, added MGF1 section.

   * Updated RSASSA-PSS section.

   * Added Public key algorithm OIDs.

   * Populated Introduction and IANA sections.

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

   * Initial version

2. Introduction

   This document describes cryptographic algorithm identifiers for
   several cryptographic algorithms which use variable length output
   SHAKE functions introduced in [SHA3] which can be used with the
   Internet X.509 Certificate and CRL profile [RFC5280].

   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 defined as SHAKE(M, d) where the output
   is a d-bits long digest of message M. 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 as the message digest function (to hash the
   message to be signed) in RSASSA-PSS and ECDSA and as the hash in the
   mask generating function in RSASSA-PSS. This specification describes
   the identifiers for SHAKEs to be used in X.509 and their meaning.

3. 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.

4. Identifiers

   This section defines four new object identifiers (OIDs), for RSASSA-
   PSS and ECDSA with each of SHAKE-128 and SHAKE-256. The same
   algorithm identifiers can be used for identifying a public key in
   RSASSA-PSS.

The new identifiers for RSASSA-PSS signatures using SHAKEs are below.

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

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

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

The new algorithm identifiers of ECDSA signatures using SHAKEs are below.

```
id-ecdsa-with-shake128 OBJECT IDENTIFIER  ::=  { joint-iso-itu-t(2)
         country(16) us(840) organization(1) gov(101)
         csor(3) algorithms(4) id-ecdsa-with-shake(3)
         TBD }



id-ecdsa-with-shake256 OBJECT IDENTIFIER  ::=  { joint-iso-itu-t(2)
         country(16) us(840) organization(1) gov(101)
         csor(3) algorithms(4) id-ecdsa-with-shake(3)
         TBD }

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

The parameters for the four identifiers above MUST be absent. That
is, the identifier SHALL be a SEQUENCE of one component, the OID.

Section 5.1.1 and Section 5.1.2 specify the required output length
for each use of SHAKE128 or SHAKE256 in RSASSA-PSS and ECDSA. In
summary, when hashing messages to be signed, output lengths of
SHAKE128 and SHAKE256 are 256 and 512 bits respectively. When the
SHAKEs are used as mask generation functions RSASSA-PSS, their output
length is (n - 264) or (n - 520) bits respectively, where n is the
RSA modulus size in bits.

5. Use in PKIX

5.1. Signatures

Signatures are used in a number of different ASN.1 structures. In an
X.509 certificate a signature is encoded with an algorithm identifier
in the signatureAlgorithm attribute and a signatureValue that
contains the actual signature.

```
   Certificate  ::=  SEQUENCE  {
       tbsCertificate       TBSCertificate,
       signatureAlgorithm   AlgorithmIdentifier,
       signatureValue       BIT STRING  }
```

The identifiers defined in Section 4 can be used as the
AlgorithmIdentifier in the signatureAlgorithm field in the sequence
Certificate and the signature field in the sequence tbsCertificate in
X.509 [RFC5280]. The parameters of these signature algorithms are
absent as explained in Section 4.

Conforming CA implementations MUST specify the algorithms explicitly
by using the OIDs specified in Section 4 when encoding RSASSA-PSS or
ECDSA with SHAKE signatures in certificates and CRLs. Conforming
client implementations that process RSASSA-PSS or ECDSA with SHAKE
signatures when processing certificates and CRLs MUST recognize the
corresponding OIDs. Encoding rules for RSASSA-PSS and ECDSA
signature values are specified in [RFC4055] and [RFC5480]
respectively.

5.1.1. RSASSA-PSS Signatures

The RSASSA-PSS algorithm is defined in [RFC8017]. When id-RSASSA-
PSS-SHAKE128 or id-RSASSA-PSS-SHAKE256 specified in Section 4 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. [RFC4055] defines RSASSA-
PSS-params that are used to define the algorithms and inputs to the
algorithm. This specification does not use parameters because the
hash and mask generating algorithsm and trailer and salt are embedded
in the OID definition.

The hash algorithm to hash a message being signed 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]. As
explained in Step 9 of section 9.1.1 of [RFC8017], the output length

of the MGF is emLen - hLen - 1 bytes. emLen is the maximum message
length ceil((n-1)/8), where n is the RSA modulus in bits. hLen is 32
and 64-bytes for id-RSASSA-PSS-SHAKE128 and id-RSASSA-PSS-SHAKE256
respectively. Thus when SHAKE is used as the MGF, the SHAKE output
length maskLen is (n - 264) or (n - 520) bits respectively. For
example, when RSA modulus n is 2048, the output length of SHAKE128 or
SHAKE256 as the MGF will be 1784 or 1528-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].

5.1.2. 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 4) algorithm identifier appears, the respective
SHAKE function (SHAKE128 or SHAKE256) 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 length of the hash function must be explicitly determined.
The output length, d, for SHAKE128 or SHAKE256 used in ECDSA MUST be
256 or 512 bits respectively.

It is RECOMMENDED that conforming CA implementations that generate
ECDSA with SHAKE signatures in certificates or CRLs generate such
signatures with a deterministically 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. These standards may
have not specified SHAKE128 and SHAKE256 as hash algorithm options.
However, SHAKE128 and SHAKE256 with output length being 32 and 64
octets respectively are subtitutions for 256 and 512-bit output hash
algorithms such as SHA256 and SHA512 used in the standards.

5.2. Public Keys

Certificates conforming to [RFC5280] can convey a public key for any
public key algorithm. The certificate indicates the public key
algorithm through an algorithm identifier. This algorithm identifier
is an OID and optionally associated parameters. The conventions and
encoding for RSASSA-PSS and ECDSA public keys algorithm identifiers

are as specified in Section 2.3 of [RFC3279], Section 3.1 of
[RFC4055] and Section 2.1 of [RFC5480].

Traditionally, the rsaEncryption object identifier is used to
identify RSA public keys. The rsaEncryption object identifier
continues to identify the subject public key when the RSA private key
owner does not wish to limit the use of the public key exclusively to
RSASSA-PSS with SHAKEs. When the RSA private key owner wishes to
limit the use of the public key exclusively to RSASSA-PSS with
SHAKEs, the AlgorithmIdentifiers for RSASSA-PSS defined in Section 4
SHOULD be used as the algorithm field in the SubjectPublicKeyInfo
sequence [RFC5280]. Conforming client implementations that process
RSASSA-PSS with SHAKE public keys when processing certificates and
CRLs MUST recognize the corresponding OIDs.

Conforming CA implementations MUST specify the X.509 public key
algorithm explicitly by using the OIDs specified in Section 4 when
encoding ECDSA with SHAKE public keys in certificates and CRLs.
Conforming client implementations that process ECDSA with SHAKE
public keys when processing certificates and CRLs MUST recognize the
corresponding OIDs.

The identifier parameters, as explained in section Section 4, MUST be
absent.

6. IANA Considerations

One object identifier for the ASN.1 module in Appendix A was assigned
in the SMI Security for PKIX Module Identifiers (1.3.6.1.5.5.7.0)
registry:

   PKIXAlgsForSHAKE-2019 { iso(1) identified-organization(3) dod(6)
      internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
      id-mod-pkix1-shakes-2019(TBD) }

7. Security Considerations

The SHAKEs are deterministic functions. Like any other deterministic
function, executing multiple times with the same input will produce
the same output. Therefore, users should not expect unrelated
outputs (with the same or different output lengths) from running a
SHAKE function with the same input multiple times. The shorter of
any two 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.

When using ECDSA with SHAKEs, the ECDSA curve order SHOULD be chosen in line with the SHAKE output length. NIST has defined appropriate use of the hash functions in terms of the algorithm strengths and expected time frames for secure use in Special Publications (SPs) [SP800-78-4] and [SP800-107]. These documents can be used as guides to choose appropriate key sizes for various security scenarios. In the context of this document id-ecdsa-with-shake128 is RECOMMENDED for curves with group order of 256-bits. id-ecdsa-with-shake256 is RECOMMENDED for curves with group order of 384-bits or more.

8. Acknowledgements

We would like to thank Sean Turner, Jim Schaad and Eric Rescorla for their valuable contributions to this document.

The authors would like to thank Russ Housley for his guidance and very valuable contributions with the ASN.1 module.

9. References

9.1. 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>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S.,
           Housley, R., and W. Polk, "Internet X.509 Public Key
           Infrastructure Certificate and Certificate Revocation List
           (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008,
           <https://www.rfc-editor.org/info/rfc5280>.

[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>.

   [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 (NIST),
              "SHA-3 Standard - Permutation-Based Hash and Extendable-
              Output Functions FIPS PUB 202", August 2015,
              <https://www.nist.gov/publications/sha-3-standard-
              permutation-based-hash-and-extendable-output-functions>.

9.2. Informative References

   [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>.

   [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>.

   [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>.

   [SP800-107]
              National Institute of Standards and Technology (NIST),
              "SP800-107: Recommendation for Applications Using Approved
              Hash Algorithms", May 2014,
              <https://csrc.nist.gov/csrc/media/publications/sp/800-107/
              rev-1/final/documents/draft_revised_sp800-107.pdf>.

   [SP800-78-4]
              National Institute of Standards and Technology (NIST),
              "SP800-78-4: Cryptographic Algorithms and Key Sizes for
              Personal Identity Verification", May 2014,
              <https://csrc.nist.gov/csrc/media/publications/sp/800-
              78/4/final/documents/sp800_78-4_revised_draft.pdf>.

   [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 module for SHAKEs in X.509. This
   module does not come from any existing RFC.

```
 PKIXAlgsForSHAKE-2019 { iso(1) identified-organization(3) dod(6)
   internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
   id-mod-pkix1-shakes-2019(TBD) }

 DEFINITIONS EXPLICIT TAGS ::=

 BEGIN

 -- EXPORTS ALL;

 IMPORTS

 -- FROM [RFC5912]

 PUBLIC-KEY, SIGNATURE-ALGORITHM, DIGEST-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) }

 -- FROM [RFC5912]

 RSAPublicKey, rsaEncryption, pk-rsa, pk-ec,
 CURVE, id-ecPublicKey, ECPoint, ECParameters, ECDSA-Sig-Value
 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-)
```

```
      --
      DigestAlgorithms DIGEST-ALGORITHM ::= {
        -- This expands DigestAlgorithms from [RFC5912]
        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 (pk-) Algorithms
      --
      PublicKeys PUBLIC-KEY ::= {
        -- This expands PublicKeys from [RFC5912]
        pk-rsaSSA-PSS-SHAKE128 |
        pk-rsaSSA-PSS-SHAKE256,
        ...
      }

      -- The hashAlgorithm is mda-shake128
      -- The maskGenAlgorithm is id-shake128
      -- Mask Gen Algorithm is SHAKE128 with output length
      -- (n - 264) bits, where n is the RSA modulus in bits.
      -- the saltLength is 32
      -- the trailerField is 1
      pk-rsaSSA-PSS-SHAKE128 PUBLIC-KEY ::= {
        IDENTIFIER id-RSASSA-PSS-SHAKE128
```

```
      KEY RSAPublicKey
      PARAMS ARE absent
      -- Private key format not in this module --
      CERT-KEY-USAGE { nonRepudiation, digitalSignature,
                       keyCertSign, cRLSign }
    }

    -- The hashAlgorithm is mda-shake256
    -- The maskGenAlgorithm is id-shake256
    -- Mask Gen Algorithm is SHAKE256 with output length
    -- (n - 520)-bits, where n is the RSA modulus in bits.
    -- the saltLength is 64
    -- the trailerField is 1
    pk-rsaSSA-PSS-SHAKE256 PUBLIC-KEY ::= {
      IDENTIFIER id-RSASSA-PSS-SHAKE256
      KEY RSAPublicKey
      PARAMS ARE absent
      -- Private key format not in this module --
      CERT-KEY-USAGE { nonRepudiation, digitalSignature,
                       keyCertSign, cRLSign }
    }

    --
    -- Signature Algorithms (sa-)
    --
    SignatureAlgs SIGNATURE-ALGORITHM ::= {
      -- This expands SignatureAlgorithms from [RFC5912]
      sa-rsassapssWithSHAKE128 |
      sa-rsassapssWithSHAKE256 |
      sa-ecdsaWithSHAKE128 |
      sa-ecdsaWithSHAKE256,
      ...
    }

    --
    -- SMIME Capabilities (sa-)
    --
    SMimeCaps SMIME-CAPS ::= {
      -- The expands SMimeCaps from [RFC5912]
      sa-rsassapssWithSHAKE128.&smimeCaps |
      sa-rsassapssWithSHAKE256.&smimeCaps |
      sa-ecdsaWithSHAKE128.&smimeCaps |
      sa-ecdsaWithSHAKE256.&smimeCaps,
      ...
    }

    -- RSASSA-PSS with SHAKE128
    sa-rsassapssWithSHAKE128 SIGNATURE-ALGORITHM ::= {
```

```
      IDENTIFIER id-RSASSA-PSS-SHAKE128
      PARAMS ARE absent
          -- The hashAlgorithm is mda-shake128
          -- The maskGenAlgorithm is id-shake128
          -- Mask Gen Algorithm is SHAKE128 with output length
          -- (n - 264) bits, where n is the RSA modulus in bits.
          -- the saltLength is 32
          -- the trailerField is 1
      HASHES { mda-shake128 }
      PUBLIC-KEYS { pk-rsa | pk-rsaSSA-PSS-SHAKE128 }
      SMIME-CAPS { IDENTIFIED BY id-RSASSA-PSS-SHAKE128 }
    }
    id-RSASSA-PSS-SHAKE128  OBJECT IDENTIFIER  ::=  { TBD }

    -- RSASSA-PSS with SHAKE256
    sa-rsassapssWithSHAKE256 SIGNATURE-ALGORITHM ::= {
      IDENTIFIER id-RSASSA-PSS-SHAKE256
      PARAMS ARE absent
          -- The hashAlgorithm is mda-shake256
          -- The maskGenAlgorithm is id-shake256
          -- Mask Gen Algorithm is SHAKE256 with output length
          -- (n - 520)-bits, where n is the RSA modulus in bits.
          -- the saltLength is 64
          -- the trailerField is 1
     HASHES { mda-shake256 }
     PUBLIC-KEYS { pk-rsa | pk-rsaSSA-PSS-SHAKE256 }
     SMIME-CAPS { IDENTIFIED BY id-RSASSA-PSS-SHAKE256 }
    }
    id-RSASSA-PSS-SHAKE256  OBJECT IDENTIFIER  ::=  { TBD }

    -- Determinstic ECDSA with SHAKE128
    sa-ecdsaWithSHAKE128 SIGNATURE-ALGORITHM ::= {
      IDENTIFIER id-ecdsa-with-shake128
      VALUE ECDSA-Sig-Value
      PARAMS ARE absent
      HASHES { mda-shake128 }
      PUBLIC-KEYS { pk-ec }
      SMIME-CAPS { IDENTIFIED BY id-ecdsa-with-shake128 }
    }
    id-ecdsa-with-shake128 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
                              country(16) us(840) organization(1)
                              gov(101) csor(3) nistAlgorithm(4)
                              sigAlgs(3) TBD }

    -- Determinstic ECDSA with SHAKE256
    sa-ecdsaWithSHAKE256 SIGNATURE-ALGORITHM ::= {
      IDENTIFIER id-ecdsa-with-shake256
      VALUE ECDSA-Sig-Value
```

```
      PARAMS ARE absent
      HASHES { mda-shake256 }
      PUBLIC-KEYS { pk-ec }
      SMIME-CAPS { IDENTIFIED BY id-ecdsa-with-shake256 }
     }
     id-ecdsa-with-shake256 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
                                 country(16) us(840) organization(1)
                                 gov(101) csor(3) nistAlgorithm(4)
                                 sigAlgs(3) TBD }

     END
```

Authors' Addresses

   Panos Kampanakis
   Cisco Systems

   Email: pkampana@cisco.com

   Quynh Dang
   NIST
   100 Bureau Drive, Stop 8930
   Gaithersburg, MD 20899-8930
   USA

   Email: quynh.dang@nist.gov