RFC 8702
Use of the SHAKE One-Way Hash Functions in the Cryptographic Message Syntax (CMS)

Abstract
This document updates the “Cryptographic Message Syntax (CMS) Algorithms” (RFC 3370) and describes the conventions for using the SHAKE family of hash functions in the Cryptographic Message Syntax as one-way hash functions with the RSA Probabilistic Signature Scheme (RSASSA-PSS) and Elliptic Curve Digital Signature Algorithm (ECDSA). The conventions for the associated signer public keys in CMS are also described.

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/rfc8702.

Copyright Notice
Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions
with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
1. Introduction

"Cryptographic Message Syntax (CMS)" [RFC5652] describes syntax used to digitally sign, digest, authenticate, or encrypt arbitrary message contents. "Cryptographic Message Syntax (CMS) Algorithms" [RFC3370] defines the use of common cryptographic algorithms with CMS. This specification updates RFC 3370 and 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 RSA Probabilistic Signature Scheme (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
defined as SHAKE(M, d) where the output is a d-bit-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 (see Appendix A.1 of [SHA3]). And the corresponding collision and
second-preimage-resistance strengths for SHAKE256 are min(d/2, 256) and min(d, 256) bits,
respectively. In this specification, we use d=256 (for SHAKE128) and d=512 (for SHAKE256).

A SHAKE can be used in CMS as the message digest function (to hash the message to be signed) in
RSASSA-PSS and ECDSA, as the message authentication code, and as the mask generation
function (MGF) in RSASSA-PSS. This specification describes the identifiers for SHAKEs to be used
in CMS and their meanings.

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.

2. Identifiers

This section identifies eight new object identifiers (OIDs) for using SHAKE128 and SHAKE256 in
CMS.

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

```plaintext
id-shake128 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
country(16) us(840) organization(1) gov(101) csor(3)
nistAlgorithm(4) 2 11 }

id-shake256 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
country(16) us(840) organization(1) gov(101) csor(3)
nistAlgorithm(4) 2 12 }
```

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

[RFC8692] defines two identifiers for RSASSA-PSS signatures using SHAKEs, which we include
here for convenience.

Kampanakis & Dang | Standards Track | Page 4
The same RSASSA-PSS algorithm identifiers can be used for identifying public keys and signatures.

[RFC8692] also defines two algorithm identifiers of ECDSA signatures using SHAKEs, which we include here for convenience.

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

In [shake-nist-oids], the National Institute of Standards and Technology (NIST) defines two object identifiers for Keccak message authentication codes (KMACs) using SHAKE128 and SHAKE256, and we include them here for convenience.

The parameters for id-KmacWithSHAKE128 and id-KmacWithSHAKE256 are **OPTIONAL**.

Sections 3.1, 3.2.1, 3.2.2, and 3.4 specify the required output length for each use of SHAKE128 or SHAKE256 in message digests, RSASSA-PSS, ECDSA, and KMAC.
3. Use in CMS

3.1. Message Digests

The id-shake128 and id-shake256 OIDs (see Section 2) can be used as the digest algorithm identifiers located in the SignedData, SignerInfo, DigestedData, and the AuthenticatedData digestAlgorithm fields in CMS [RFC5652]. The OID encoding MUST omit the parameters field and the output length of SHAKE128 or SHAKE256 as the message digest MUST be 32 or 64 bytes, respectively.

The digest values are located in the DigestedData field and the Message Digest authenticated attribute included in the signedAttributes of the SignedData signerInfos. In addition, digest values are input to signature algorithms. The digest algorithm MUST be the same as the message hash algorithms used in signatures.

3.2. Signatures

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

Conforming implementations that process RSASSA-PSS and ECDSA with SHAKE signatures when processing CMS data MUST recognize the corresponding OIDs specified in Section 2.

When using RSASSA-PSS or ECDSA with SHAKEs, the RSA modulus or ECDSA curve order SHOULD be chosen in line with the SHAKE output length. Refer to Section 5 for more details.

3.2.1. RSASSA-PSS Signatures

The RSASSA-PSS algorithm is defined in [RFC8017]. When id-RSASSA-PSS-SHAKE128 or id-RSASSA-PSS-SHAKE256 (specified in Section 2) 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, mask generation algorithm, trailer, and salt are embedded in the OID definition.

The hash algorithm used to hash a message being signed and the hash algorithm as the mask generation function used in RSASSA-PSS MUST be the same: both SHAKE128 or both SHAKE256. The output length of the hash algorithm that hashes the message SHALL be 32 (for SHAKE128) or 64 bytes (for SHAKE256).

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, instead of the MGF1 algorithm that uses the hash function in multiple iterations, as specified in Appendix B.2.1 of [RFC8017]. In other words, the MGF is defined as the SHAKE128 or SHAKE256 with input being the mgfSeed for id-RSASSA-PSS-
SHAKE128 and id-RSASSA-PSS-SHAKE256, respectively. The mgfSeed is an octet string used as the seed to generate the mask [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 (8*emLen - 264) or (8*emLen - 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 bytes for id-RSASSA-PSS-SHAKE128 or 64 bytes for id-RSASSA-PSS-SHAKE256. Finally, the trailerField MUST be 1, which represents the trailer field with hexadecimal value 0xBC [RFC8017].

3.2.2. 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 2) 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 [X9.62], the output length of the hash function must be explicitly determined. The output length for SHAKE128 or SHAKE256 used in ECDSA MUST be 32 or 64 bytes, respectively.

Conforming Certification Authority (CA) implementations that generate ECDSA with SHAKE signatures in certificates or Certificate Revocation Lists (CRLs) SHOULD generate such signatures with a deterministically generated, nonrandom 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 those standards. Those standards have not specified SHAKE128 and SHAKE256 as hash algorithm options. However, SHAKE128 and SHAKE256 with output length being 32 and 64 octets, respectively, can be used instead of 256 and 512-bit output hash algorithms, such as SHA256 and SHA512.

3.3. Public Keys

In CMS, the signer's public key algorithm identifiers are located in the OriginatorPublicKey's algorithm attribute. 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 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, the AlgorithmIdentifier for RSASSA-PSS defined in Section 2 SHOULD be used as the
algorithm attribute in the OriginatorPublicKey sequence. Conforming client implementations that process RSASSA-PSS with SHAKE public keys in CMS message MUST recognize the corresponding OIDs in Section 2.

Conforming implementations MUST specify and process the algorithms explicitly by using the OIDs specified in Section 2 when encoding ECDSA with SHAKE public keys in CMS messages. The identifier parameters, as explained in Section 2, MUST be absent.

3.4. Message Authentication Codes

Keccak 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 OID 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 32 or 64 bytes, 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.

4. IANA Considerations

One object identifier for the ASN.1 module in Appendix A was updated in the "Structure of Management Information (SMI) Security for S/MIME Module Identifier (1.2.840.113549.1.9.16.0)" registry:

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Description</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>CMSAlgsForSHAKE-2019</td>
<td>RFC 8702</td>
</tr>
</tbody>
</table>

Table 1

5. Security Considerations

This document updates [RFC3370]. The security considerations section of that document applies to this specification as well.

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.
SHAKE128 with an output length of 32 bytes offers 128 bits of collision and preimage resistance. Thus, SHAKE128 OIDs in this specification are **RECOMMENDED** with a 2048- (112-bit security) or 3072-bit (128-bit security) RSA modulus or curves with a group order of 256 bits (128-bit security). SHAKE256 with a 64-byte output length offers 256 bits of collision and preimage resistance. Thus, the SHAKE256 OIDs in this specification are **RECOMMENDED** with 4096-bit RSA modulus or higher or curves with group order of at least 512 bits, such as NIST curve P-521 (256-bit security). Note that we recommended a 4096-bit RSA because we would need a 15360-bit modulus for 256 bits of security, which is impractical for today's technology.

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.

6. References

6.1. Normative References


6.2. Informative References


[SP800-107] National Institute of Standards and Technology (NIST), "Recommendation for Applications Using Approved Hash Algorithms", Draft NIST Special Publication


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-2019 { iso(1) member-body(2) us(840)  
    rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) modules(0)  
    id-mod-cms-shakes-2019(70) }  

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) }  

sa-rsassapssWithSHAKE128, sa-rsassapssWithSHAKE256,  
sa-ecdsaWithSHAKE128, sa-ecdsaWithSHAKE256  
FROM 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(94) } ;  

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

-- SHAKE256  
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 signed-data content
-- type and countersignature attribute in CMS.
--
-- From RFC 5280, 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 ::= { iso(1)
--  identified-organization(3) dod(6) internet(1)
--  security(5) mechanisms(5) pkix(7) algorithms(6) 30 }

id-RSASSA-PSS-SHAKE256 OBJECT IDENTIFIER ::= { iso(1)
  identified-organization(3) dod(6) internet(1)
  security(5) mechanisms(5) pkix(7) algorithms(6) 31 }

-- When the id-RSASSA-PSS-* algorithm identifiers are used
-- for a public key or 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 output length, respectively. The mask generation
-- function MUST be SHAKE128 or SHAKE256 with an output length
-- of (8*ceil((n-1)/8) - 264) or (8*ceil((n-1)/8) - 520) bits,
-- respectively, where n is the RSA modulus in bits.
-- The RSASSA-PSS saltLength MUST be 32 or 64 bytes, respectively.
-- The trailerField MUST be 1, which represents the trailer
-- field with hexadecimal value 0xBC. Regardless of
-- id-RSASSA-PSS-* or rsaEncryption being used as the
-- AlgorithmIdentifier of the OriginatorPublicKey, the RSA
-- public key MUST be encoded using the RSAPublicKey type.

-- From RFC 4055, for reference:
-- RSAPublicKey ::= SEQUENCE {
--   modulus INTEGER, -- -- n
--   publicExponent INTEGER } -- -- e

id-ecdsa-with-shake128 OBJECT IDENTIFIER ::= { iso(1)
  identified-organization(3) dod(6) internet(1)
  security(5) mechanisms(5) pkix(7) algorithms(6) 32 }

id-ecdsa-with-shake256 OBJECT IDENTIFIER ::= { iso(1)
  identified-organization(3) dod(6) internet(1)
  security(5) mechanisms(5) pkix(7) algorithms(6) 33 }

-- When the id-ecdsa-with-shake* algorithm identifiers are
-- used in CMS, the AlgorithmIdentifier parameters field
-- MUST be absent and the signature algorithm should be
deterministic ECDSA [RFC6979]. The message digest MUST
be SHAKE128 or SHAKE256 with a 32- or 64-byte output
length, respectively. In both cases, the ECDSA public key,
MUST be encoded using the id-ecPublicKey type.

-- From RFC 5480, 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
-- are defined as:
-- ECPublicKey ::= CHOICE {
--   namedCurve OBJECT IDENTIFIER
--   -- -- implicitCurve NULL
--   -- -- specifiedCurve SpecifiedECDomain
--   }

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

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

-- This expands SMimeCaps from [RFC5911]
--
SMimeCaps SMIME-CAPS ::= {
  -- sa-rassapssWithSHAKE128.&smimeCaps |
  -- sa-rassapssWithSHAKE256.&smimeCaps |
  -- sa-ecdsaWithSHAKE128.&smimeCaps |
  -- sa-ecdsaWithSHAKE256.&smimeCaps |
  maca-KMACwithSHAKE128.&smimeCaps |
  maca-KMACwithSHAKE256.&smimeCaps,
  ...
}

--
-- 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.
Acknowledgements

This document is based on Russ Housley's document [CMS-SHA3]. It replaces SHA3 hash functions by SHAKE128 and SHAKE256, as the LAMPS WG agreed.

The authors would like to thank Russ Housley for his guidance and very valuable contributions with the ASN.1 module. Valuable feedback was also provided by Eric Rescorla.

Authors' Addresses

Panos Kampanakis
Cisco Systems
Email: pkampana@cisco.com