LAMPS WG
Internet-Draft
Intended status: Standards Track
Expires: December 31, 2018
Q. Dang
NIST
P. Kampanakis
Cisco Systems
June 29, 2018

Use of the SHAKE One-way Hash Functions in the Cryptographic Message Syntax (CMS)
draft-ietf-lamps-cms-shakes-01

Abstract

This document describes the conventions for using the SHAKE family of hash functions with the Cryptographic Message Syntax (CMS).

Status of This Memo

Copyright Notice

Table of Contents

1. Change Log

[EDNOTE: Remove this section before publication.]

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 MFG, 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.

The SHA-3 family of one-way hash functions is specified in [SHA3]. In the SHA-3 family, two extendable-output functions, called SHAKE128 and SHAKE256 are defined. Four hash functions, 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 lengths, in bits, of the SHAKE hash functions are defined by the d parameter. The corresponding collision and preimage resistance security levels for SHAKE128 and SHAKE256 are respectively min(d/2,128) and min(d,128) and min(d/2,256) and min(d,256) bits.

A SHAKE can be used in CMS as a message digest, message authentication code or a mask generation function (in RSASSA-PSS). In this document we define six new OIDs using SHAKE128 and SHAKE256 in CMS.

3. Identifiers

The 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.

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) 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 same RSASSA-PSS and ECDSA with SHAKEs algorithm identifiers are used for identifying public keys and signatures.

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.

The new object identifiers for KMACs using SHAKE128 and SHAKE256 are below.

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

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

EDNOTE: "TBD" will be specified by NIST.

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

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.

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 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 algorithm in the maskGenAlgorithm used in RSASSA-PSS MUST be the same, SHAKE128 or SHAKE256 respectively. The output-length of the SHAKE which hashes the message SHALL be 32 or 64 bytes respectively.

The maskGenAlgorithm is the MGF1 specified in Section B.2.1 of [RFC8017]. A mask generation function in RSASSA-PSS takes an octet string of variable length and a desired output length as input, and outputs an octet string of the desired length. The output length for SHAKE128 or SHAKE256 being used as the hash function in MGF1 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 for SHAKE128 or SHAKE256 in the maskGenAlgorithm will be 223 or 191 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. 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. The ECDSA message hash function is SHAKE128 or SHAKE256 respectively.

4.3. Public Keys

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

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

When id-ecdsa-with-shake128 or id-ecdsa-with-shake256 are used as the algorithm identitifier in the public key, the parameters, as explained in Section 3, MUST be absent. The hash function and its output-length are the same as in Section 4.2.2.

Additionally, 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 KMAC algorithm identifier, the parameters field MUST be absent.

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

This document uses several new registries [EDNOTE: Update here.]

6. Security Considerations

SHAKE128 and SHAKE256 are one-way extensible-output functions. Their output length depends on a required length of the consuming application.

The SHAKEs are deterministic functions. Like any other deterministic functions, 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.

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.

Implementations must randomly generate message-authentication keys and one-time values, such as the k value when generating a ECDSA signature. In addition, the generation of public/private key pairs relies on random numbers. The use of inadequate pseudo-random number generators (PRNGs) to generate such cryptographic values can result in little or no security. The generation of quality random numbers is difficult. [RFC4086] offers important guidance in this area, and [SP800-90A] series provide acceptable PRNGs.

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

[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>.

[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>.

[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>.

[SP800-90A] National Institute of Standards and Technology, "Recommendation for Random Number Generation Using Deterministic Random Bit Generators. NIST SP 800-90A", June 2015, <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.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

[EDNOTE: Update]

Authors' Addresses

Quynh Dang
NIST
100 Bureau Drive
Gaithersburg, MD 20899

Email: quynh.Dang@nist.gov

Panos Kampanakis
Cisco Systems

Email: pkampana@cisco.com