Internet-Draft Q. Dang
Intended status: Standards Track NIST
Expires: 29 April 2018 P. Kampanakis
Cisco Systems
29 October 2017

# Use of the SHAKE One-way Hash Functions in the Cryptographic Message Syntax (CMS)

<draft-dang-lamps-cms-shakes-hash-00.txt>

## Abstract

This document describes the conventions for using 2 one-way hash functions called SHAKE128 and SHAKE256 in the SHA3 family with the Cryptographic Message Syntax (CMS).

## Status of This Memo



## Copyright Notice

# 1. Introduction

The Cryptographic Message Syntax (CMS) [CMS] 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 2 new hash funcitons with the CMS. In addition, this specification describes the use of these 2 one-way hash functions with the RSASSA PKCS#1 version 1.5 signature algorithm [PKCS1] and the Elliptic Curve Digital Signature Algorithm (ECDSA) [DSS] with the CMS signed-data content type.

## 1.1. ASN.1

CMS values are generated using ASN.1 [ASN1-B], using the Basic Encoding Rules (BER) and the Distinguished Encoding Rules (DER) [ASN1-E].

## 1.2. 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 RFC 2119 [KEYWORDS].

# 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 SHAKE128 and SHAKE256 [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.

Output lengths of SHAKE128 and SHAKE256 are always 256 and 512 bits respectively in this specification. The object identifiers for these 2 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-SHAKE128 OBJECT IDENTIFIER ::= { hashAlgs 11 }
```

```
id-SHAKE256 OBJECT IDENTIFIER ::= { hashAlgs 12 }
```

When using the id-SHAKE128 or id-SHAKE256 algorithm identifier, the parameters field MUST be absent; not NULL but absent. Again, the output lengths are fixed as 256 and 512 bits respectively.

## 3. Signature Algorithms

This section specifies the conventions employed by CMS implementations that support 2 SHAKE one-way hash functions with the RSASSA PKCS#1 version 1.5 signature algorithm [PKCS1] and the Elliptic Curve Digital Signature Algorithm (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 SHAKEs

The RSASSA PKCS#1 v1.5 is defined in [PKCS1]. When RSASSA PKCS#1 v1.5 is used in conjunction with one of the SHAKEs 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-rsassa-pkcs1-v1_5-with-SHAKE128 ::= { sigAlgs x }

id-rsassa-pkcs1-v1_5-with-SHAKE256 ::= { sigAlgs y }
```

Note: x and y will be specified by NIST.

The algorithm identifier for RSASSA PKCS#1 v1.5 subject public keys in certificates is specified in [PKIXALG], 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-SHAKE128 or id-rsassa-pkcs1-v1_5-with-SHAKE256 algorithm identifier is used, 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 [PKIXALG]. 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, and that value is used directly as the signature value.

## 3.2. ECDSA with SHAKEs

The Elliptic Curve Digital Signature Algorithm (ECDSA) is defined in [DSS]. When ECDSA is used in conjunction with one of the SHAKE 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-SHAKE128 ::= { sigAlgs x }

id-ecdsa-with-SHAKE256 ::= { sigAlgs y }
```

Note: x and y will be specified by NIST.

When using the id-ecdsa-with-SHAKE128 or id-ecdsa-with-SHAKE256 algorithm identifier, the parameters field MUST be absent; not NULL but absent.

The conventions for ECDSA public keys is as specified in [PKIXECC]. 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 [PKIXALG] and repeated here for convenience:

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

## 4. Message Authentication Codes with SHAKEs

This section specifies the conventions employed by CMS implementations that support the KMAC specified in [KMAC] as authentication code (MAC).

KMAC algorithm identifiers are located in the AuthenticatedData macAlgorithm field.

MAC values are located in the AuthenticatedData mac field.

The object identifiers for KMACs with SHAKE128 and SHAKE256 are:

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

id-KmacWithSHAKE128 OBJECT IDENTIFIER ::= { hashAlgs x }

id-KmacWithSHAKE256 OBJECT IDENTIFIER ::= { hashAlgs y }
```

Note: x and y will be specified by NIST.

The variables N and S in this specification for KMAC are emply strings. L, an integer representing the requested output length in bits, is 256 or 512 for KmacWithSHAKE128 or KmacWithSHAKE256 respectively in this specification.

When the id-KmacWithSHAKE128 or id-KmacWithSHAKE256 algorithm identifier is used, the parameters field MUST be absent; not NULL but absent.

## 5. Security Considerations

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. RFC 4086 [RANDOM] offers important guidance in this area, and NIST SP 800-90 [SP800-90s] series provide acceptable PRNGs.

Implementers should be aware that cryptographic algorithms may 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.

## 6. Normative References

[ASN1-B] ITU-T, "Information technology -- Abstract Syntax Notation One (ASN.1): Specification of basic notation", ITU-T Recommendation X.680, 2015.

[ASN1-E] 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, 2015.

[CMS] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, September 2009.

[DSS] National Institute of Standards and Technology, U.S. Department of Commerce, "Digital Signature Standard, version 4", NIST FIPS PUB 186-4, 2013.

[HMAC] Krawczyk, H., "HMAC: Keyed-Hashing for Message Authentication", RFC 2104. February 1997.

[KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[PKCS1] Moriarty, K., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2" RFC 8017, November 2016.

[PKIXALG] 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, April 2002.

[PKIXECC] Turner, S., Brown, D., Yiu, K., Housley, R., and T. Polk, "Elliptic Curve Cryptography Subject Public Key Information", RFC 5480, March 2009.

[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.

[SP800-90s] National Institute of Standards and Technology, SP 800-90A,B & C.

## 7. Informative References

[RANDOM] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

## Appendix A ASN.1 Module

TBD

## Appendix B Acknowledgement

This document is just an update of Russ Housley’s draft:
https://tools.ietf.org/html/draft-housley-lamps-cms-sha3-hash-00
This document replaced SHA3 hash functions by SHAKE128 and SHAKE256
as the LAMPS working group agreed.

## Authors’ Addresses

Quynh Dang & Kampanakis
NIST
100 Bureau Drive
Gaithersburg, MD 20899

Email: quynh.Dang & Kampanakis@nist.gov

Panos Kampanakis
Cisco Systems

Email: pkampana@cisco.com