LAMPS WG   Q. Dang
Internet-Draft   NIST
Intended status: Standards Track   P. Kampanakis
Expires: August 19, 2018   Cisco Systems
                                                       February 15, 2018

Use of the SHAKE One-way Hash Functions in the Cryptographic Message Syntax (CMS)
draft-ietf-lamps-cms-shakes-00

Abstract

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

Table of Contents

1. Change Log

[EDNOTE: Remove this section before publication.]

- 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, this specification describes the use of these one-way hash 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.

3. Message Digest Algorithms

3.1. One-way Extensible-Output-Function SHAKEs

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.

In CMS, 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.

SHAKE is a variable length hash function. The output lengths, in bits, of the SHAKE hash functions is defined by the parameter d. 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). The Object Identifiers (OIDs) for these two hash functions are defined in [shake-nist-oids] and are included 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) hashalgs(2) 17 }

id-shake128-len OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
                 country(16) us(840) organization(1) gov(101) csor(3)
                 nistalgorithm(4) hashalgs(2) 18 }

ShakeOutputLen ::= INTEGER -- Output length in octets
```

When using the id-shake128-len id-shake256-len algorithm identifiers, the parameters MUST be present, and they MUST employ the ShakeOutputLen syntax that contains an encoded positive integer value at least 32 or 64 respectively.

3.2. Mask Generation SHAKEs

The RSASSA-PSS signature algorithm uses a mask generation function. A 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. The mask generation function used in RSASSA-PSS

is defined in [RFC8017], but we include it here as well for convenience:

```
id-mgf1  OBJECT IDENTIFIER  ::=  { pkcs-1 8 }
```

The parameters field associated with id-mgf1 MUST have a hashAlgorithm value that identifies the hash used with MGF1. To use SHAKE as this hash, this parameter MUST be id-shake128-len or id-shake256-len as specified in Section 3.1 above.

4. Signature Algorithms

This section specifies the conventions employed by CMS implementations that support 2 SHAKE one-way hash 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 CMS, signature algorithm identifiers are located in the SignerInfo signatureAlgorithm field of SignedData and countersignature attributes. Signature values are located in the SignerInfo signature field of SignedData and countersignature attributes.

4.1. RSASSA-PSS with SHAKEs

The RSASSA-PSS signature algorithm identifier and its parameters are specifed in [RFC4055]:

```
id-RSASSA-PSS  OBJECT IDENTIFIER  ::=  { pkcs-1 10 }

RSASSA-PSS-params  ::=  SEQUENCE  {
     hashAlgorithm      HashAlgorithm,
     maskGenAlgorithm   MaskGenAlgorithm,
     saltLength         INTEGER,
     trailerField       INTEGER }
```

This document adds two new hash algorithm choices and two new choices for mask generation functions. These are the SHAKE128 and SHAKE256 algorithm identifiers specified in Section 3.1.

When SHAKE128 or SHAKE256 is used as the hashAlgorithm, it MUST also be used as the maskGenAlgorithm.

When used as the hashAlgorithm, the SHAKE128 or SHAKE256 output-length must be either 32 or 64 bytes respectively. In these cases, the parameters MUST be present, and they MUST employ the ShakeOutputLen syntax that contains an encoded positive integer value

of 32 or 64 for id-shake128-len or id-shake256-len algorithm identifier respectively.

When id-shake128-len or id-shake256-len algorithm identifier is used as the id-mfg1 maskGenAlgorithm parameter, the ShakeOutputLen parameter must be (n - 264)/8 or (n - 520)/8 respectively for SHAKE128 and SHAKE256, where n is the RSA modulus in bits. For example, when RSA modulus n is 2048, ShakeOutputLen must be 223 or 191 when id-shake128-len or id-shake256-len is used respectively.

The parameter saltLength MUST be 32 or 64 bytes respectively for the SHAKE128 and SHAKE256 OIDs.

The conventions for RSA public keys are as specified in [RFC3279] and [RFC4055]. [RFC3279] defines the following OID for RSA with NULL parameters.

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

Additionally, [RFC4055] adds the RSASSA-PSS OID and parameters shown above as a public key identifier. The parameters may be either absent or present when RSASSA-PSS OID is used as subject public key information. If id-RSASSA-PSS is used in the public key identifier with parameters, Section 3.3 of [RFC4055] describes that the signature algorithm parameters MUST match the parameters in the key structure algorithm identifier except the saltLength field. The saltLength field in the signature parameters MUST be greater or equal to that in the key parameters field. If the id-RSASSA-PSS parameters are NULL no further parameter validation is necessary.

4.2. ECDSA with SHAKEs

The Elliptic Curve Digital Signature Algorithm (ECDSA) is defined in [X9.62]. When ECDSA is used in conjunction with one of the SHAKE one-way hash functions, the object identifiers are:

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

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

EDNOTE: 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.

For simplicity and compliance with the ECDSA standard specification, the output size of the hash function must be explicitly determined. The ShakeOutputLen parameter of SHAKE128 or SHAKE256 MUST be 32 or 64 bytes respectively when it is used in ECDSA

The conventions for ECDSA public keys is specified in [RFC5480] as

```
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.

5. Message Authentication Codes with SHAKEs

This section specifies the conventions employed by CMS implementations that support the KMAC specified in [SP800-185] as authentication code (MAC).

In CMS, 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:

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

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

EDNOTE: z and w will be specified by NIST.

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

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.

6. Acknowledgement

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.

7. IANA Considerations

This document uses several registries that were originally created in [shake-nist-oids]. No further registries are required. [EDNOTE: Update here.]

8. 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 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.

9. References

9.1. Normative 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>.

[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>.

9.2. 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.

[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