

LAMPS WG
Internet-Draft
Intended status: Standards Track
Expires: August 19, 2018

P. Kampanakis
Cisco Systems
Q. Dang
NIST
February 15, 2018

Internet X.509 Public Key Infrastructure: Additional SHAKE Algorithms
and Identifiers for RSA and ECDSA
draft-ietf-lamps-pkix-shake-01

Abstract

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

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 19, 2018.

Copyright Notice

Copyright (c) 2018 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

Internet-Draft

SHAKE identifiers in X.509

February 2018

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.

Table of Contents

1.	Change Log	2
2.	Introduction	3
3.	Message Digest Algorithms	3
3.1.	One-way Extensible-Output-Function SHAKEs	3
3.2.	Mask Generation SHAKEs	4
4.	Signature Algorithms	4
4.1.	RSASSA-PSS with SHAKEs	4
4.2.	ECDSA with SHAKEs	5
5.	Public Key Algorithms	6
6.	Acknowledgements	7
7.	IANA Considerations	7
8.	Security Considerations	7
9.	References	8
9.1.	Normative References	8
9.2.	Informative References	9
Appendix A.	ASN.1 module	9
	Authors' Addresses	9

[1.](#) Change Log

[EDNOTE: Remove this section before publication.]

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](#):

[2.](#) Introduction

This document describes several cryptographic algorithms which may be used with the Internet X.509 Certificate and CRL profile [[RFC5280](#)]. It describes the OIDs for variable length SHAKE algorithms introduced in [[SHA3](#)] and how they can be used in X.509 certificates. [EDNOTE: Update here.]

[3.](#) Message Digest Algorithms

This section describes two one-way hash functions and digital signature algorithms using these functions, which may be used to sign certificates and CRLs, and identifies OIDs (Object Identifiers) for public keys contained in certificates.

[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. 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) hashalg(2) 17 }
```

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

When using the id-shake128-len algorithm identifier, the parameters

MUST be present, and they MUST employ the ShakeOutputLen syntax that contains an encoded positive integer value at least 32 in this specification.

```
id-shake256-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-shake256-len algorithm identifier, the parameters MUST be present, and they MUST employ the ShakeOutputLen syntax that

contains an encoded positive integer value at least 64 in this specification.

[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

[4.1.](#) RSASSA-PSS with SHAKES

The RSASSA-PSS signature algorithm identifier and its parameters are specified 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 are used respectively.

The parameter saltLength MUST be 32 or 64 bytes respectively for the SHAKE128 and SHA256 OIDs.

[4.2](#). ECDSA with SHAKEs

The Elliptic Curve Digital Signature Algorithm (ECDSA) is defined in "Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Standard (ECDSA)" [[X9.62](#)]. The ASN.1 OIDs of ECDSA signature algorithms using SHAKE128 and SHAKE256, are below:

```
id-ecdsa-with-shake128 OBJECT IDENTIFIER ::= { joint-iso-ccitt(2)
country(16) us(840) organization(1) gov(101) csor(3) al
```

id-ecdsa-with-shake(3) x }

id-ecdsa-with-shake256 OBJECT IDENTIFIER ::= { joint-iso-ccitt(2)
country(16) us(840) organization(1) gov(101) csor(3) al
id-ecdsa-with-shake(3) y }

[EDNOTE: "x" and "y" will be specified by NIST later.]

When the id-ecdsa-with-SHAKE128 or id-ecdsa-with-SHAKE256, algorithm identifier appears in the algorithm field as an AlgorithmIdentifier, the encoding MUST omit the parameters field. That is, the AlgorithmIdentifier SHALL be a SEQUENCE of one component, the OID ecdsa-with-SHAKE128 or ecdsa-with-SHAKE256.

Conforming CA implementations MUST specify the hash algorithm explicitly using the OIDs specified in [Section 3.2](#) above when encoding ECDSA/SHAKE signatures in certificates and CRLs.

Conforming client implementations that process ECDSA signatures with any of the SHAKE hash algorithms when processing certificates and CRLs MUST recognize the corresponding OIDs specified in Sections [3.1](#) and [3.2](#) above.

Encoding rules for ECDSA signature values are specified in [\[RFC4055\]](#), [Section 2.2.3](#), and [\[RFC5480\]](#).

Conforming CA implementations that generate ECDSA signatures in certificates or CRLs MUST generate such ECDSA signatures in accordance with all the requirements specified in Sections [7.2](#) and [7.3](#) of [\[X9.62\]](#) or with all the requirements specified in Section 4.1.3 of [\[SEC1\]](#). They MAY also generate such ECDSA signatures in accordance with all the recommendations in [\[X9.62\]](#) or [\[SEC1\]](#) if they have a stated policy that requires conformance to these standards. These standards above 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 substitutions for 256 and 512-bit output hash algorithms such as SHA256 and SHA512 used in the standards.

5. Public Key Algorithms

The conventions for RSA and ECDSA public keys are as specified in [\[RFC3279\]](#), [\[RFC4055\]](#) and [\[RFC5480\]](#). We include them here for convenience.

[\[RFC3279\]](#) defines the following OID for RSA with NULL parameters.

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

Additionally, [\[RFC4055\]](#) adds the corresponding RSASSA-PSS OID public key identifier and parameters (also shown in [Section 4](#) of this document). The parameters may be either absent or present when RSASSA-PSS OID is used as subject public key information.

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

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.

For ECDSA, [\[RFC5480\]](#) defines the EC public key identifier and its parameters 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 signer's certificate SHALL apply to the verification of the signature.

[6.](#) Acknowledgements

We would like to thank Sean Turner for his valuable contributions to this document.

[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>>.
- [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>>.

- [SHA3] National Institute of Standards and Technology, "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

- [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>>.
- [SEC1] Standards for Efficient Cryptography Group, "SEC 1: Elliptic Curve Cryptography", May 2009, <<http://www.secg.org/sec1-v2.pdf>>.
- [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: More here.]

Authors' Addresses

Panos Kampanakis
Cisco Systems

Email: pkampana@cisco.com

Internet-Draft

SHAKE identifiers in X.509

February 2018

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

Email: quynh.dang@nist.gov

