INTERNET-DRAFT
Intended Status: Proposed Standard
Expires: 30 November 2018
R. Housley
Vigil Security
30 May 2018

Use of the Hash-based Signature Algorithm with CBOR Object Signing and Encryption (COSE)

<draft-housley-suit-cose-hash-sig-00>

Abstract

This document specifies the conventions for using the Leighton-Micali Signature (LMS) algorithm for digital signatures with the CBOR Object Signing and Encryption (COSE) syntax.

Copyright and License Notice

Table of Contents

1. Introduction

This document specifies the conventions for using the Leighton-Micali Signature (LMS) algorithm [HASHSIG] for digital signatures with the CBOR Object Signing and Encryption (COSE) [RFC8152] syntax. The LMS algorithm is one form of hash-based digital signature; it can only be used for a fixed number of signatures. The LMS algorithm uses small private and public keys, and it has low computational cost; however, the signatures are quite large.

Commented [DT1]: It would also be useful to add to this section a brief summary of what problem this document is trying to solve. For example, one paragraph worded in laymans terms. I like the last paragraph of section 5.2. Maybe put that, or some variation of it, before this paragraph, and in the abstract.

2. 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.

3. LMS Digital Signature Algorithm Overview

This specification makes use of the hash-based signature algorithm specified in [HASHSIG], which is the Leighton and Micali adaptation [LM] of the original Lamport-Diffie-Winternitz-Merkle one-time signature system [M1979][M1987][M1989a][M1989b].

Commented [DT2]: What is the status of this normative reference? It seems to still be an individual draft, and the filename does not imply it's submitted to any WG.

The hash-based signature algorithm has three major components:

- o Hierarchical Signature System (HSS) -- see Section 3.1;
- o Leighton-Micali Signature (LMS) -- see Section 3.2; and
- o Leighton-Micali One-time Signature Algorithm (LM-OTS) -- see Section 3.3.

As implied by the name, the hash-based signature algorithm depends on a collision-resistant hash function, and this specification makes use of the SHA-256 one-way hash function [SHS].

3.1. Hierarchical Signature System (HSS)

The hash-based signature algorithm specified in [HASHSIG] uses a hierarchy of trees. The Hierarchical Signature System (HSS) allows subordinate trees to be generated when needed by the signer. Otherwise, generation of the entire tree might take weeks or longer.

Commented [DT3]: I don't understand this sentence, please reword/clarify.

An HSS signature as specified in [HASHSIG] carries the number of signed public keys (Nspk), followed by that number of signed public keys, followed by the LMS signature as described in Section 3.2.

Each signed public key is represented by the hash value at the root of the tree, and the signature over that public key is an LMS signature as described in Section 3.2.

The elements of the HSS signature value for a stand-alone tree can be summarized as:

```
u32str(0) ||
lms_signature_on_message
```

The elements of the HSS signature value for a tree with Nspk levels can be summarized as:

```
u32str(Nspk) ||
lms_signature_on_public_key[0] || public_key[1] ||
lms_signature_on_public_key[1] || public_key[2] ||
   ...
lms_signature_on_public_key[Nspk-2] || public_key[Nspk-1] ||
lms_signature_on_public_key[Nspk-1] || public_key[Nspk] ||
lms_signature_on_message
```

3.2. Leighton-Micali Signature (LMS)

Each tree in the hash-based signature algorithm specified in [HASHSIG] uses the Leighton-Micali Signature (LMS) system. LMS systems have two parameters. The first parameter is the height of the tree, h, which is the number of levels in the tree minus one. The hash-based signature algorithm supports five values for this parameter: h=5; h=10; h=15; h=20; and h=25. Note that there are 2^h leaves in the tree. The second parameter is the number of bytes output by the hash function, m, which is the amount of data associated with each node in the tree. This specification supports only SHA-256, with m=32.

The hash-based signature algorithm supports five tree sizes:

```
LMS_SHA256_M32_H5;
LMS_SHA256_M32_H10;
LMS_SHA256_M32_H15;
LMS_SHA256_M32_H20; and
LMS_SHA256_M32_H25.
```

An LMS signature consists of four elements: a typecode indicating the particular LMS algorithm, the number of the leaf associated with the LM-OTS signature, an LM-OTS signature as described in Section 3.3, and an array of values that is associated with the path through the tree from the leaf associated with the LM-OTS signature to the root. The array of values contains the siblings of the nodes on the path from the leaf to the root but does not contain the nodes on the path itself. The array for a tree with height h will have h values. The first value is the sibling of the leaf, the next value is the sibling of the parent of the leaf, and so on up the path to the root.

The four elements of the LMS signature value can be summarized as:

```
u32str(q) ||
ots_signature ||
u32str(type) ||
path[0] || path[1] || ... || path[h-1]
```

3.3. Leighton-Micali One-time Signature Algorithm (LM-OTS)

The hash-based signature algorithm depends on a one-time signature method. This specification makes use of the Leighton-Micali One-time Signature Algorithm (LM-OTS) [HASHSIG]. An LM-OTS has five parameters:

n - The number of bytes output by the hash function. This specification supports only SHA-256 [SHS], with n=32.

H - A preimage-resistant hash function that accepts byte strings of any length, and returns an n-byte string. This specification supports only SHA-256 [SHS].

w - The width in bits of the Winternitz coefficients. [HASHSIG] supports four values for this parameter: w=1; w=2; w=4; and w=8.

p - The number of n-byte string elements that make up the LM-OTS signature.

ls - The number of left-shift bits used in the checksum function, which is defined in Section 4.5 of [HASHSIG].

The values of p and ls are dependent on the choices of the parameters n and w, as described in Appendix A of [HASHSIG].

The hash-based signature algorithm supports four LM-OTS variants:

```
LMOTS_SHA256_N32_W1;
LMOTS_SHA256_N32_W2;
LMOTS_SHA256_N32_W4; and
LMOTS_SHA256_N32_W8.
```

Signing involves the generation of C, which is an n-byte random value.

The LM-OTS signature value can be summarized as:

```
u32str(type) || C || y[0] || ... || y[p-1]
```

4. Hash-based Signature Algorithm Identifiers

The CBOR Object Signing and Encryption (COSE) [RFC8152] supports two signature algorithm schemes. This specification makes use of the signature with appendix scheme for hash-based signatures.

The signature value is a large byte string. The byte string is designed for easy parsing, and it includes a counter and type codes that indirectly provide all of the information that is needed to parse the byte string during signature validation. The first four bytes of the signature value contains the number of signed public keys (Nspk) in the HSS. The first four bytes of each LMS signature value contains type code, which tells how to parse the remaining parts of the LMS signature value. The first four bytes of each LM-OTS signature value contains type code, which tells how to parse the remaining parts of the LM-OTS signature value.

When using a COSE key for this algorithm, the following checks are made:

- The 'kty' field MUST be present, and it MUST be 'HSIG'.
- If the 'alg' field is present, and it MUST be 'HSIG'.
- If the 'key_ops' field is present, it MUST include 'sign' when creating a hash-based signature.
- If the 'key_ops' field is present, it MUST include 'verify' when verifying a hash-based signature.

5. Security Considerations

5.1. Implementation Security Considerations

Implementations must protect the private keys. Compromise of the private keys may result in the ability to forge signatures. Along with the private key, the implementation must keep track of which leaf nodes in the tree have been used. Loss of integrity of this tracking data can cause an one-time key to be used more than once. As a result, when a private key and the tracking data are stored on non-volatile media or stored in a virtual machine environment, care must be taken to preserve confidentiality and integrity.

An implementation must ensure that a LM-OTS private key is used to

Commented [DT4]: Do you want to recommend say TPM, or mention it as an implementation example?

Commented [DT5]: Grammar: just “a”

generate a signature only one time, and ensure that it cannot be used for any other purpose.

The generation of private keys relies on random numbers. The use of inadequate pseudo-random number generators (PRNGs) to generate these values can result in little or no security. An attacker may find it much easier to reproduce the PRNG environment that produced the keys, searching the resulting small set of possibilities, rather than brute force searching the whole key space. The generation of quality random numbers is difficult. [RFC4086] offers important guidance in this area.

5.2. Algorithm Security Considerations

At Black Hat USA 2013, some researchers gave a presentation on the current sate of public key cryptography. They said: "Current cryptosystems depend on discrete logarithm and factoring which has seen some major new developments in the past 6 months" [BH2013]. They encouraged preparation for a day when RSA and DSA cannot be depended upon.

A post-quantum cryptosystem is a system that is secure against quantum computers that have more than a trivial number of quantum bits. It is open to conjecture when it will be feasible to build such a machine. RSA, DSA, and ECDSA are not post-quantum secure.

The LM-OTP one-time signature, LMS, and HSS do not depend on discrete logarithm or factoring, as a result these algorithms are considered to be post-quantum secure.

Today, RSA is often used to digitally sign software updates. This means that the distribution of software updates could be compromised if a significant advance is made in factoring or a quantum computer is invented. The use of hash-based signatures to protect software update distribution will allow the deployment of software that implements new cryptosystems.

6. Operational Considerations

The public key for the hash-based signature is the ke at the root of Hierarchical Signature System (HSS). In the absence of a public key infrastructure [RFC5280], this public key is a trust anchor, and the number of signatures that can be generated is bounded by the size of the overall HSS set of trees. When all of the LM-OTS signatures have been used to produce a signature, then the establishment of a new trust anchor is required.

Commented [DT6]: typo

To ensure that none of tree nodes are used to generate more than one signature, the signer maintains state across different invocations of the signing algorithm. Section 12.2 of [HASHSIG] offers some practical implementation approaches around this statefulness. In some of these approaches, nodes are sacrificed to ensure that none are used more than once. As a result, the total number of signatures that can be generated might be less than the overall HSS set of trees.

7. IANA Considerations

IANA is requested to add entries for hash-based signatures in the "COSE Algorithms" registry and hash-based public keys in the "COSE Key Types" registry.

7.1. COSE Algorithms Registry Entry

The new entry in the "COSE Algorithms" registry has the following columns:

Name: HASHSIG

Value: TBD (Value to be assigned by IANA)

Description: Hash-based digital signatures

Reference: This document (Number to be assigned by RFC Editor)

Recommended: Yes

7.2. COSE Key Types Registry Entry

The new entry in the "COSE Key Types" registry has the following columns:

Name: HASHSIG

Value: TBD (Value to be assigned by IANA)

Description: Hash-based digital signature public key

Reference: This document (Number to be assigned by RFC Editor)

8. References

8.1. Normative References

[HASHSIG] McGrew, D., M. Curcio, and S. Fluhrer, "Hash-Based Signatures", Work in progress. <draft-mcgrew-hash-sigs-11>

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", RFC 8152, DOI 10.17487/RFC8152, July 2017, <https://www.rfc-editor.org/info/rfc8152>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[SHS] National Institute of Standards and Technology (NIST), FIPS Publication 180-3: Secure Hash Standard, October 2008.

8.2. Informative References

[BH2013] Ptacek, T., T. Ritter, J. Samuel, and A. Stamos, "The Factoring Dead: Preparing for the Cryptopocalypse", August 2013. <https://media.blackhat.com/us-13/us-13-Stamos-The-Factoring-Dead.pdf>

[LM] Leighton, T. and S. Micali, "Large provably fast and secure digital signature schemes from secure hash functions", U.S. Patent 5,432,852, July 1995.

[M1979] Merkle, R., "Secrecy, Authentication, and Public Key Systems", Stanford University Information Systems Laboratory Technical Report 1979-1, 1979.

[M1987] Merkle, R., "A Digital Signature Based on a Conventional Encryption Function", Lecture Notes in Computer Science crypto87, 1988.

[M1989a] Merkle, R., "A Certified Digital Signature", Lecture Notes in Computer Science crypto89, 1990.

[M1989b] Merkle, R., "One Way Hash Functions and DES", Lecture Notes in Computer Science crypto89, 1990.

[PQC] Bernstein, D., "Introduction to post-quantum cryptography", 2009. <http://www.pqcrypto.org/www.springer.com/cda/content/document/cda_downloaddocument/9783540887010-c1.pdf>

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <http://www.rfc-editor.org/info/rfc4086>.

[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>.

Author's Address

Russ Housley
Vigil Security, LLC
918 Spring Knoll Drive
Herndon, VA 20170
USA

EMail: housley@vigilsec.com