Abstract

This document specifies the conventions for using the Hierarchical Signature System (HSS) / Leighton-Micali Signature (LMS) hash-based signature algorithm with the CBOR Object Signing and Encryption (COSE) syntax. The HSS/LMS algorithm is one form of hash-based digital signature; it is described in RFC 8554.

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 May 6, 2020.

Copyright Notice

Copyright (c) 2019 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 include Simplified BSD License text as described in Section 4.e of
1. Introduction

This document specifies the conventions for using the Hierarchical Signature System (HSS) / Leighton-Micali Signature (LMS) hash-based signature algorithm with with the CBOR Object Signing and Encryption (COSE) [RFC8152] syntax. The LMS system provides a one-time digital signature that is a variant of Merkle Tree Signatures (MTS). The HSS is built on top of the LMS system to efficiently scale for a larger numbers of signatures. The HSS/LMS algorithm is one form of hash-based digital signature, and it is described in [HASHSIG]. The HSS/LMS signature algorithm can only be used for a fixed number of signing operations. The number of signing operations depends upon the size of the tree. The HSS/LMS signature algorithm uses small public keys, and it has low computational cost; however, the signatures are quite large. The HSS/LMS private key can be very small when the signer is willing to perform additional computation at signing time; alternatively, the private key can consume additional memory and provide a faster signing time. The HSS/LMS signatures [HASHSIG] are currently defined to use exclusively SHA-256 [SHS].
1.1. Motivation

Recent advances in cryptanalysis [BH2013] and progress in the development of quantum computers [NAS2019] pose a threat to widely deployed digital signature algorithms. As a result, there is a need to prepare for a day that cryptosystems such as RSA and DSA that depend on discrete logarithm and factoring cannot be depended upon.

If large-scale quantum computers are ever built, these computers will be able to break many of the public-key cryptosystems currently in use. A post-quantum cryptosystem [PQC] is a system that is secure against quantum computers that have more than a trivial number of quantum bits (qubits). It is open to conjecture when it will be feasible to build such computers; however, RSA, DSA, ECDSA, and EdDSA are all vulnerable if large-scale quantum computers come to pass.

Since the HSS/LMS signature algorithm does not depend on the difficulty of discrete logarithm or factoring, the HSS/LMS signature algorithm is considered to be post-quantum secure. The use of HSS/LMS hash-based signatures to protect software update distribution, perhaps using the format that is being specified by the IETF SUIT Working Group, will allow the deployment of software that implements new cryptosystems.

1.2. 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] when, and only when, they appear in all capitals, as shown here.

2. LMS Digital Signature Algorithm Overview

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

The hash-based signature algorithm has three major components:

- Hierarchical Signature System (HSS) -- see Section 2.1;
- Leighton-Micali Signature (LMS) -- see Section 2.2; and
- Leighton-Micali One-time Signature Algorithm (LM-OTS) -- see Section 2.3.
As implied by the name, the hash-based signature algorithm depends on a collision-resistant hash function. The hash-based signature algorithm specified in [HASHSIG] currently makes use of the SHA-256 one-way hash function [SHS], but it also establishes an IANA registry to permit the registration of additional one-way hash functions in the future.

2.1. Hierarchical Signature System (HSS)

The hash-based signature algorithm specified in [HASHSIG] uses a hierarchy of trees. The Hierarchical N-time 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.

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 2.2. The public key for the top-most LMS tree is the public key of the HSS system. The LMS private key in the parent tree signs the LMS public key in the child tree, and the LMS private key in the bottom-most tree signs the actual message. The signature over the public key and the signature over the actual message are LMS signatures as described in Section 2.2.

The elements of the HSS signature value for a stand-alone tree (a top tree with no children) can be summarized as:

```
u32str(0) ||
  lms_signature /* signature of message */
```

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

```
u32str(Nspk) ||
  signed_public_key[0] ||
  signed_public_key[1] ||
  ...
  signed_public_key[Nspk-2] ||
  signed_public_key[Nspk-1] ||
  lms_signature /* signature of message */
```

where, as defined in Section 3.3 of [HASHSIG], a signed_public_key is the lms_signature over the public key followed by the public key itself. Note that Nspk is the number of levels in the hierarchy of trees minus 1.
2.2. 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 [HASHSIG] includes support for five values of 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 $m=32$. An IANA registry is defined so that other hash functions could be used in the future.

The [HASHSIG] specification 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.

The [HASHSIG] specification establishes an IANA registry to permit the registration of additional hash functions and additional tree sizes in the future.

The [HASHSIG] specification defines the value $I$ as the private key identifier, and the same $I$ value is used for all computations with the same LMS tree. In addition, the [HASHSIG] specification defines the value $T[i]$ as the $m$-byte string associated with the $i$th node in the LMS tree, where and the nodes are indexed from 1 to $2^{h+1}-1$. Thus, $T[1]$ is the $m$-byte string associated with the root of the LMS tree.

The LMS public key can be summarized as:

```
   u32str(lms_algorithm_type) || u32str(otstype) || I || T[1]
```

As specified in [HASHSIG], the LMS signature consists of four elements: the number of the leaf associated with the LM-OTS signature, an LM-OTS signature as described in Section 2.3, a typecode indicating the particular LMS algorithm, 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]
```

## 2.3. 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 B of [HASHSIG].

The [HASHSIG] specification supports four LM-OTS variants:

- LMOTS_SHA256_N32_W1;
- LMOTS_SHA256_N32_W2;
- LMOTS_SHA256_N32_W4; and
- LMOTS_SHA256_N32_W8.

The [HASHSIG] specification establishes an IANA registry to permit the registration of additional hash functions and additional parameter sets in the future.
Signing involves the generation of C, which is an n-byte random value.

The LM-OTS signature value can be summarized as the identifier of the LM-OTS variant, the random value, and a sequence of hash values (y[0] through y[p-1]) that correspond to the elements of the public key as described in Section 4.5 of [HASHSIG]:

\[
u32str(\text{otstype}) \ || \ C \ || \ y[0] \ || \ ... \ || \ y[p-1]
\]

3. 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 as described in Section 2. The byte string is designed for easy parsing. The HSS, LMS, and LMOTS components of the signature value format include counters and type codes that indirectly provide all of the information that is needed to parse the byte string during signature validation.

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

- The 'kty' field MUST be present, and it MUST be 'HSS-LMS'.
- If the 'alg' field is present, and it MUST be 'HSS-LMS'.
- 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.
- If the 'kid' field is present, it MAY be used to identify the top of the HSS tree. In [HASHSIG], this identifier is called 'I', and it is the 16-byte identifier of the LMS public key for the tree.

4. Security Considerations

4.1. 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 a 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, failed writes, virtual machine snapshotting or cloning, and other operational concerns must be considered to ensure confidentiality and integrity.

When generating an LMS key pair, an implementation MUST generate each key pair independently of all other key pairs in the HSS tree.

An implementation MUST ensure that a LM-OTS private key is used to 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, and [RFC4086] offers important guidance in this area.

The generation of hash-based signatures also depends on random numbers. While the consequences of an inadequate pseudo-random number generator (PRNG) to generate these values is much less severe than in the generation of private keys, the guidance in [RFC4086] remains important.

5. Operational Considerations

The public key for the hash-based signature is the key 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.

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.
6. 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.

6.1. COSE Algorithms Registry Entry

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

Name: HSS-LMS
Value: TBD (Value between -256 and 255 to be assigned by IANA)
Description: HSS/LMS hash-based digital signature
Reference: This document (Number to be assigned by RFC Editor)
Recommended: Yes

6.2. COSE Key Types Registry Entry

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

Name: HSS-LMS
Value: TBD (Value to be assigned by IANA)
Description: Public key for HSS/LMS hash-based digital signature
Reference: This document (Number to be assigned by RFC Editor)

7. References

7.1. Normative References


7.2. Informative References


Appendix A. Examples

This appendix provides a non-normative example of a COSE full message signature and an example of a COSE_Sign1 message. This section follows the formatting used in [RFC8152].

The programs that were used to generate the examples can be found at https://github.com/cose-wg/Examples.

A.1. Example COSE Full Message Signature

This section provides an example of a COSE full message signature.

Size of binary file is 2560 bytes.

98{
    / protected / h’a10300’ / {
        \ content type \ 3:0
    } /,
    / unprotected / {},
    / payload / ‘This is the content.’,
    / signatures / [ ]
    / protected / h’a101382d’ / {
        \ alg \ 1:-46 \ HSS-LMS \ }
    / unprotected / {
        / kid / 4:’ItsBig’
    },
    / signature / h’000000000000000100000391291de76ce6e24d1e2a
9b60266519bc8ce889f8174eb0fc00edd3129de3ab9b6bfa3bf47d007d84af7db74
9ea97215e82f456cbdd473812c6a042ae39539898752c89b60a276ec8a9feab900e2
5bdfe0ab8e773a1c36ae214d67c655bb68630450a5db2c7c6403b77f6a9bf4d30a02
19db5cdec884d7514f3cbda9220020bf3045b0e5c6955b32864f16f97da02f0cbf6a
70458b7032e30b0342d75b8f3dc6871442e6384b10f559f5dc594a214924c48ccc3
37078665653fc740340428138b0fb515f4f2f2cb291ad05ace7acea6031b2d09b2f4
This section provides an example of a COSE_Sign1 message.

Size of binary file is 2552 bytes.

```
18(
   / protected / h'a101382d' / {
      \ alg \ 1:-46 \ HSS-LMS
   } ,
   / unprotected / {
      / kid / 4:’ItsBig’,
      / payload / ‘This is the content.’,
      / signature / h’00000000000000000000000391291de76ce6e24d1e2a9b60266519bc8ce889f814deb0fc00eddd3129de3a9b9a5b5ac783bd0fe689f57fb204f1992db1c2e2484f316c74bce3f2094cfae964a9548cead0f78ee5d549510d190f647320448ae27ece780a0c39c645bf8db08573af52c93d91fd0e217f245c752c176b81514eb630670e6fbb329225eaa88c7d21635e32ae84213f89018cb6f1b84e61ec348b690d7c6265c199f9868952d9982eadc417b5279dd674cd951c306016cfeef4ee3bfffce5e5a5d0bb5bf4f3bc93995f26cfe7c0c15b25741cf2d8470939e8bd47ef9b9cf309ef895226e92be60683459099611deffbb9a43217956a0ab2959b
   }
)`

A.2. Example COSE_Sign1 Message

This section provides an example of a COSE_Sign1 message.
Internet-Draft          HSS/LMS HashSig with COSE          November 2019

Housley                    Expires May 6, 2020                 

Page 14
Appendix B.  Acknowledgements

Many thanks to Roman Danyliw, Scott Fluhrer, Laurence Lundblade, John Mattsson, Jim Schaad, and Tony Putman for their valuable review and insights.  In addition, an extra special thank you to Jim Schaad for generating the examples in Appendix A.

Author’s Address

Russ Housley
Vigil Security, LLC
516 Dranesville Road
Herndon, VA  20170
US

Email: housley@vigilsec.com