RFC 8778
Use of the HSS/LMS Hash-Based Signature Algorithm with CBOR Object Signing and Encryption (COSE)

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 is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8778.

Copyright Notice
Copyright (c) 2020 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 the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
# Table of Contents

1. Introduction  
   1.1. Motivation  
   1.2. Terminology  
2. LMS Digital Signature Algorithm Overview  
   2.1. Hierarchical Signature System (HSS)  
   2.2. Leighton-Micali Signature (LMS)  
   2.3. Leighton-Micali One-Time Signature (LM-OTS) Algorithm  
3. Hash-Based Signature Algorithm Identifiers  
4. Security Considerations  
5. Operational Considerations  
6. IANA Considerations  
   6.1. COSE Algorithms Registry Entry  
   6.2. COSE Key Types Registry Entry  
   6.3. COSE Key Type Parameters Registry Entry  
7. References  
   7.1. Normative References  
   7.2. Informative References  
Appendix A. Examples  
   A.1. Example COSE Full Message Signature  
   A.2. Example COSE_Sign1 Message  
Acknowledgements  
Author's Address
1. Introduction

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) [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 number of signatures. The HSS/LMS algorithm is one form of a 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 have more than a trivial number of quantum bits (qubits), and they 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 such large-scale quantum computers. When it will be feasible to build such computers is open to conjecture; however, RSA [RFC8017], DSA [DSS], Elliptic Curve Digital Signature Algorithm (ECDSA) [DSS], and Edwards-curve Digital Signature Algorithm (EdDSA) [RFC8032] 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 will allow the deployment of future software that implements new cryptosystems. By deploying HSS/LMS today, authentication and integrity protection of the future software can be provided, even if advances break current digital-signature mechanisms.

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] [RFC8174] 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 Micali 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
- Leighton-Micali One-time Signature (LM-OTS) Algorithm-- 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 N-time 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.

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 topmost 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 */
```

where the notation comes from [HASHSIG].

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

Subordinate LMS trees are placed in the HSS structure, as discussed in Section 2.1. 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. The [HASHSIG] 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
- 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. The value I is also available in the public key. In addition, the [HASHSIG] specification defines the value T[r] as the m-byte string associated with the ith node in the LMS tree, 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 type code 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 (LM-OTS) Algorithm

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

- **n:** The number of bytes output by the hash function. For SHA-256 [SHS], n=32.
- **H:** A preimage-resistant hash function that accepts byte strings of any length and returns an
  n-byte string.
- **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 4.4
  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**
- **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]), as described in Section 4.5 of [HASHSIG]:

\[ u32str(otstype) \mid C \mid y[0] \mid \ldots \mid y[p-1] \]

### 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 LM-OTS 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 'HSS-LMS'.
  - If the 'alg' field is present, 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

The security considerations from [RFC8152] and [HASHSIG] are relevant to implementations of this specification.

There are a number of security considerations that need to be taken into account by implementers of this specification.

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 nonvolatile media or 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 an 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 pseudorandom 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 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 the tree nodes are used to generate more than one signature, the signer maintains state across different invocations of the signing algorithm. Section 9.2 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.

A COSE Key Type Parameter for encoding the HSS/LMS private key and the state about which tree nodes have been used is deliberately not defined. It was not defined to avoid creating the ability to save the private key and state, generate one or more signatures, and then restore the private key and state. Such a restoration operation provides disastrous opportunities for tree node reuse.

## 6. IANA Considerations

IANA has added entries for the HSS/LMS hash-based signature algorithm in the "COSE Algorithms" registry and added HSS/LMS hash-based signature public keys in the "COSE Key Types" registry and the "COSE Key Type Parameters" registry.

### 6.1. COSE Algorithms Registry Entry

The new entry in the "COSE Algorithms" registry [IANA] appears as follows:
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 is formatted according to the extended CBOR diagnostic format defined by [RFC8610].

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.

The 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'0000000000000000000000391291de76ce6e24d1e2a9b60266519bc8ce889f814deeb0fc90edd3129de3ab9b6bfa3bf47d007d844a7f7db749ea97215e82f456cbdd473812c6a842ae39539898752c89b60a276e8a99feab900e25bdf0ab8e773aa1c36ae214d67c65bb686304505db2c7c6403b77f6a9bf4d30a0219db5cceed884d7514f3ccbd1922020bf3045b0e5c6955b32864f16f97da02f0cbfe
  ]
)
A.2. Example COSE_Sign1 Message

This section provides an example of a COSE_Sign1 message.

The size of the 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'00000000000000000000000391291de76ce6e24d1e2a9b60
        266519bc8ce889f814de8b0fc80e6dd3129de3ab9b9a5b5ac783bf60fe689f5f7b204
        f1992db1c1e2484f316c74bc3f2909cfa8e96a4a9548ceda0f78ee5d549510d1910
        f64732b44a8e27e72498920a0c39c645bfb8db868537af52c93d91fd0e217f2f45c7
        52c176b81514eb6e3067ef0fbb329225eaa88c7d216353e32a84213f78018e67f1b8
        4e61eac3486b90dc76265c19f9d8689592982acded41b5729d764c951306016
        cfeefee3b3fc5e5a5ad08b5bf453b93995f26cfe7c0c1c5b2574c1f2d8470993
        e8bd47f9c309f895226e92be60683459009611deffba943217596a0ab2959bb
        da09feca39de37e7c4a6cd8a5314d6b02b3774065a5e58e91feafa9f24e4ec186b2a1
        6f33c787449933e40a6651f7f1d3c19e38c634d9890bc580324c0bfc7c5f0aca014
        b4af20b0a73f969cddba94da86c8e80c76158d4f5cf3c2baf91f393df47e556887f91
        6854085242a05ec6cbb76595ec3dd2fedefae3fd1608a7801c2265f5f83c9b1ed3152
        ddaac7426c363990bec8f1da6174abe8d3568c976b149eb977d61ac15b8f1b18bce
        5f9f1d4e448ee81f375e1f96a52d39619459f131826143ee8899bad4085f5ef6cd3d3a2
        274316e68670c8b42c3801e1e90251bebed18e0956967158cc2774760adcc23
        c149a89ed2a52478882dcd15f2f33844535e0f21000bd5575313df4f21875680e6d5e
        97f6681f9dd9f88c9b28b94a0f237a6e3879f2b9098e618eb7d6a368729ca9a9829f91f
        be1c584c35f6e2734d53d1035bd2893a201c889a37a558b160f1bad01879a11f8
        881609e944dcde47a7ae6a828899d7761f8e9d5a5a45460882e85d6565e7fb5758
        9eabe1a8e792d745f3a66245f8a18181767274273c69e7a0735e5dad07a2e24b3817b8
        3b893212eb747e65f54a26d577a22eb8cd8c4a999f4190b0e2f29dc88689dc923995
        c1989beee35d1e6ed68665f5446df8e997b6e85f5f648415233dee3b9da82db29
        e8c3d3e59dbdb55e6348cd9f421783db998e087de4642562d513597b00672f3e172fad
        87752a79ceee8b2a38b1e6f2562836721cbbfba20f13113c009a436b93a0b44fcbb
    ]
)
```
Acknowledgements

Many thanks to Roman Danyliw, Elwyn Davies, Scott Fluhrer, Ben Kaduk, 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
United States of America
Email: housley@vigilsec.com