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,
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]:

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

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 'T', 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'0000000000000000000000000000000001291de76ce6e24d1e2a9b60266519bc8ce889f814de0bfc090edd3129de3ab9b6bfa3bf47d007d844af47db749e97215e82456cbdd4738126a042ae39539898752c89b66a276e8a9fefa9b00ee5bdfae8e773a1c36ae214d67c65bb6863045045db2c7ce6403b77f6a9bf4430a0219db5cced884d7514f3c605b0e5c6955b32864f16f97da20f0c6fe
```
A.2. Example COSE_Sign1 Message

This section provides an example of a COSE_Sign1 message.

The size of binary file is 2552 bytes.

Housley Standards Track Page 13
Housley Standards Track Page 14
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