Use of the HSS/LMS Hash-based Signature Algorithm with CBOR Object
Signing and Encryption (COSE)
draft-ietf-cose-hash-sig-04

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 April 12, 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] [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 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:

```plaintext
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:

```plaintext
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:

- \( \text{LMS}_\text{SHA256}_M32_H5 \)
- \( \text{LMS}_\text{SHA256}_M32_H10 \)
- \( \text{LMS}_\text{SHA256}_M32_H15 \)
- \( \text{LMS}_\text{SHA256}_M32_H20 \)
- \( \text{LMS}_\text{SHA256}_M32_H25 \)

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

The LMS public key can be summarized as:

\[
\text{u32str(lms_algorithm_type)} \ || \ \text{u32str(otstype)} \ || \ I \ || \ T[1]
\]

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:

\[
\text{u32str(q)} \ || \ \text{ots_signature} \ || \ \text{u32str(type)} \ || \ \text{path[0]} \ || \ \text{path[1]} \ || \ldots \ || \ \text{path[h-1]}
\]
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(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 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 an example of a COSE full message signature and an example of a COSE_Sign0 message. The display format includes "\" to indicate that the same field continues on the next line, and it includes "\" to separate items within a field.

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.

```
{  
    "title":"HSS LMS Hash based signature - hsssig-01",
    "input":{
        "plaintext":"This is the content.",
        "sign":{
            "protected":{
                "ctyp":0
            },
            "signers":{
                "key":{
                    "kty":"HSS-LMS",
                    "kid":"ItsBig",
                    "comment":"1 level key - LM_SHA256_MD32_H10 + \ 
                            LMOTS_SHA256_N32_W4 ",
                    "public":"000000010000000600000003d08fabd4a20 \
                            91ff0a8cb4ed834e7453432a58855cd9ba0431235 \
                            466bff9651c6c92124404d4f5a3cf161c28f1ad5a8e",
                    "private":"1|6|3|558B8966C48AE9CB898B423C8344 \
                            3AAE014A72F1B1AB5CC85CF1D892903B5439|1|d0 \
                            8fabd4a2091ff0a8cb4ed834e74534"
                },
                "unprotected":{
                    "kid":"ItsBig"
                },
                "protected":{
                    "alg":"HSS-LMS"
                }
            }
        }
        "rng_description":"Random value for signature",
        "rng_stream":{
            "ACFC5C7377D45C969DF7D7289882A48C1A10E5C48B6E29DF5018D \
```
3E683E36BC5" 
}
"
"intermediates":
"signers":
{
"ToBeSign_hex": "856953696766E17475726543A103004AA10 \ 1674853532D4CD53405454686973206973207468652063 \ 6F674656E742E"
}
}
"
"output":
"cbor_diag": "98 ([h'10A30100', {}, h'54686973206973207468652 \ 036E6E74656742E', [h'10A1674853532D4CD53', \ (4: h'497473426977'), h'00000000000000000091291 \ DE76CE6247E2A9B6G0626519BC8CE889F84D8EB0FCC0EDD319D \ E3A9B9F0C5F0A46923923AA3209BF9E1840A0B28906D794C9280A \ DC6300C184CB333429EC0035FE3E2E4428770D22F285687A18AAE76 \ CDC2F8E8F4043B314A6E72F9F679F7E35A34594E7674EE7B580E \ 840FBBF2DA398EC59BF023FD0D34ACEA319DC1ED1BD22B13A094 \ 6160F30168A6E193C57C32BB017C22529EC3760FF9335863D5A6 \ 97F0F805BDD720E72F758B19D4E27D114BE6321BDFD1859102E7 \ 23A3B1F1AE5BC3EC8732F2F1B2C4D384137E8EEEC94804CB47C82 \ 3CB0B1441E28B178E1F5A904CF7592AAACF820C97E7714B69FCA4 \ BABEB7854BC00A705CAE7BAA9112D182C21BCE3F10EA70C32F46 \ 6749279610A3477B03E3622169438C27CD46FCA7679D010D0B13A \ 06F5C0D093932EEB2BBOE25BFFFD2A08C8DF065318B7BFEDB34 \ 6EB56BDA75B0421DF87F7FD1F08808B58DD3647472D90F8F9459C \ 775BFFFF93905956B7D7FD5F6B26BC53196FF9B660949B2315B9C \ E7A0DF55E9083B42A9D08F281D2EF262770EBDCE424A450448B54 \ 15C7BA81BEFED2BCC8C1B69322156469EF160DD779CA4DDDF74774 \ 85BB5B01AAD4FD4D64594293574D35D3D49D8032BC8F8A719 \ D6BB5E90891C8787193C8E3E83655C5807A17CE1BFF830D4 \ B7D6D3E46E8FC1EE99071BB67A123FDEC3F37638DCAF04BF308 \ 4074069171C10C4670163B926635ADE3B67D970DB0297D4B8 \ 8B005473B6FF3862FB49C1DE1F6069B306C4EF8AE4C7F83EB320 \ A20406AA7FF48BFD22DA876B4661ED583F35591625FD53DC01 \ BD472DB4D93E93DA318CD5CEA706BD7BCFA510E5BD31C1AA60 \ BD252071D689C99D5CC1DEFE8AA235C654F758FC8936515EA3441 \ C3B9FF2AFFA164AF299C6994C54F0AC92F0E6ED84C86C18234E \ ADB87ABEA39C353BD9682E60D1215061316928474327E1E47C5 \ 9D9BB4C36E72746642B136304C611E53159775196A4C4D5 \ 9D55F553FE49597DF17873BB5E478278D3AA48BD22849D5A97C \ 93A10A637DE2DE52EB7D56D2C56CE32700513C0FC0F25649A6FC5C \ DBB5634C0BBB66E1FAADA66923D21B3BF62C6DD6D7AAB67AB \ A8923C4FC1CC8A8BB47F33DF12617C387B1DB386E262D2E3DF5A9 \ F8E7609667B717A5B492C6AE25920E8F697F7666D0222CD48528"
A.2. Example COSE_Sign0 Message

This section provides an example of a COSE_Sign0 message.

```json
{
  "title": "HSS LMS Hash based signature - hsssig-sig-01",
  "input": {
    "plaintext": "This is the content.",
    "sign0": {
      "key": {
        "kty": "HSS-LMS",
        "kid": "ItsBig",
        "comment": "1 level key - LM_SHA256_MD32_H10 + \ LMOTS_SHA256_N32_W4 ",
        "public": "00000000000000000000000000000000d08fabd4a2091ff0a \ 8c84ed834e7453432a58885cd9ba0431235466bff651c6 \ c9212440d4d4f53cf161c28f1ad5a8e",
        "private": "1|6|3|558B8966C48AE9CB89B423C834343AE01 \ 4A172F1B1A5C85CF1D892903B5439|0|d08fabd4a2091f \ f0a8cb4ed834e74534",
      },
      "unprotected": {
        "kid": "ItsBig"
      },
      "protected": {
        "alg": "HSS-LMS"
      },
      "alg": "HSS-LMS"
    }
  }
}``
"rng_description":"Random value for signature",
"rng_stream":{
  "1D5112D38A1146402875B73BC8D4B59C845C6AE61D03A70ABAD09 \ 8AC05AD8297"
},
"intermediates":{
  "ToBeSign_hex":"846A5369676E6174757265314AA10167485352D4 \ C4D534054546869732069732074686520366F674656E742E"
},
"output":{
  "cbor_diag":"18(['h'\'A10167485352D4C4D53',' \ (4: 'h'\'497473426967')],
 h'\'546869730269730274666520366F674656E742E',
 h'00000000000000000000000391291DE76CE62D1E2A9B60266
  519BC8CE889BF14DEBF0CO0ED3129DE3AB98A6814A\4BE84E5E8
  3C87725F78FE0610837A548F92802DA610AFB0ADFB131323061CO
  23E8A7A802C17B00740F2537A775B9E29390B6F0CA0A87095
  5420A680031331AE1A12083134238DFE5F163E159CFD207BC79B
  50DD39BA39FCAAA75C127F71B943AB8736162E42C2C2F9159DF3
  32C39A50BB8404F2CB6D98DAAC3DF82A197CFE01BEC72C820
  A5B26AC5DF05947E3A7D9207A0465C67B0A95BA0499AF6558
  1B719912E296765FA46CB0A02ED56BBCF00CA6FB9C16D8C05C1C1
  65FED054A099A3DA899FC9B51C6DE366DF38E299CE7DC9AC9C43
  66F328407E7C4A6C8A5314D6B02B377406D5A58E59E91FEAA9F2
  E4EC1682BA1F633C7784B3038FAC2E77947916C8F4160C69D0B
  0BC6600CDBC4AE9474DD5D317DCBA3D200A739F96CDDBA94D4AF86C
  E80C76158D45CF3CD2BA9F1393DF47E556887F919E071B625D31
  240E7FE959912F757314C20893827194AD6555F1452E3A749CE2
  13DFBA23BO31CDCCF196F9B1D8715B6E7451DE35B1818DF8A62
  DE1480F2DDC126B477E019FE75E4472EF4FA1B913C80821155
  AAAED7F3B1175B64C076926166C80EB219D241791C1DE3C8F936
  55085C0B0FBF840970367DAF2A41D462C696C74AFFC3591AC684D
  7019638199FBDC9457585D4E687B4AD086A31F6A5E174A6C63
  25CF67CE2A42DCEDE047BB201670CC2D6603DF2232F4BA3373D25
  16638161FD4738997BB3EC1499E594B5CCE9FBCC13136C20F1
  6012A4DA62675627757B8DF1BE635C87F81713D322EA1276F6C
  8880F42A5B19780CCEE2B78BA0C21E434AD7C490BE30E19CAFCA3
  A5CF97451FADF412ECA7BAD72E2553541224EB9349A9CA003EC1
  59EA2D9EAF6D6A72F1C43A07B70CBBC0AD844506E314C484F39F
  F28915239C8CB733787EE79704B8BA0CCE67282984DFBD01BC34
  ADF0E90A309986BE6AD95486E67754543999AE160A7C645892F
  9C338136D95FB66E80EF4F03D8F942875DF39E89EAA6B35FF6A
  C910975E5EA0DEA90DDAC52FBE16A830EA495D50ABEFBB6C824
  4095992BAE73D790AE908CE9413A6EC52F98F9583E1383E6B78A
  A6657E03DF715516DD9343274D222F9A4DB6759999DD01F7E7
  C1C2S45C22ASEA109519C788FA1504B0273975C3E647820CF65CDF4
  D0D5A6C7171C1795EB2E37030EDEB31965866255B1FC10BD03B0

Housley                  Expires April 12, 2020                [Page 17]
Appendix B. Acknowledgements

Many thanks to Roman Danyliw, Scott Fluhrer, 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