ï»¿

Internet Engineering Task Force (IETF)
Request for Comments: 8778
Category: Standards Track
ISSN: 2070-1721
April 2020

R. Housley
Vigil Security

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

Copyright Notice

Table of Contents

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:

```
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 */
```

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) || 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 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:

Name: HSS-LMS
Value: -46
Description: HSS/LMS hash-based digital signature
Reference: RFC 8778
Recommended: Yes

6.2. COSE Key Types Registry Entry

The new entry in the "COSE Key Types" registry [IANA] appears as follows:

Name: HSS-LMS
Value: 5
Description: Public key for HSS/LMS hash-based digital signature
Reference: RFC 8778

6.3. COSE Key Type Parameters Registry Entry

The new entry in the "COSE Key Type Parameters" registry [IANA] appears as follows:

Key Type: 5
Name: pub
Label: -1
CBOR Type: bstr
Description: Public key for HSS/LMS hash-based digital signature
Reference: RFC 8778

7. References

7.1. Normative References

[HASHSIG] McGrew, D., Curcio, M., and S. Fluhrer, "Leighton-Micali Hash-Based Signatures", RFC 8554, DOI 10.17487/RFC8554, April 2019, <https://www.rfc-editor.org/info/rfc8554>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://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), "Secure Hash Standard", FIPS Publication 180-4, DOI 10.6028/NIST.FIPS.180-4, August 2015, <https://doi.org/10.6028/NIST.FIPS.180-4>.

7.2. Informative References

[BH2013] Ptacek, T., Ritter, T., Samuel, J., 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>.

[DSS] National Institute of Standards and Technology (NIST), "Digital Signature Standard (DSS)", FIPS Publication 186-4, DOI 10.6028/NIST.FIPS.186-4, July 2013, <https://doi.org/10.6028/NIST.FIPS.186-4>.

[IANA] IANA, "CBOR Object Signing and Encryption (COSE)", <https://www.iana.org/assignments/cose>.

[LM] Leighton, F. 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", Information Systems Laboratory, Stanford University, Technical Report No. 1979-1, June 1979.

[M1987] Merkle, R., "A Digital Signature Based on a Conventional Encryption Function", Advances in Cryptology -- CRYPTO ’87 Proceedings, Lecture Notes in Computer Science, Volume 291, DOI 10.1007/3-540-48184-2_32, 1988, <https://doi.org/10.1007/3-540-48184-2_32>.

[M1989a] Merkle, R., "A Certified Digital Signature", Advances in Cryptology -- CRYPTO ’89 Proceedings, Lecture Notes in Computer Science, Volume 435, DOI 10.1007/0-387-34805-0_21, 1990, <https://doi.org/10.1007/0-387-34805-0_21>.

[M1989b] Merkle, R., "One Way Hash Functions and DES", Advances in Cryptology -- CRYPTO ’89 Proceedings, Lecture Notes in Computer Science, Volume 435, DOI 10.1007/0-387-34805-0_40, 1990, <https://doi.org/10.1007/0-387-34805-0_40>.

[NAS2019] National Academies of Sciences, Engineering, and Medicine, "Quantum Computing: Progress and Prospects", The National Academies Press, DOI 10.17226/25196, 2019, <http://dx.doi.org/10.17226/25196>.

[PQC] Bernstein, D., "Introduction to post-quantum cryptography", DOI 10.1007/978-3-540-88702-7_1, 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, <https://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>.

[RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016, <https://www.rfc-editor.org/info/rfc8017>.

[RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <https://www.rfc-editor.org/info/rfc8032>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/info/rfc8610>.

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’00000000000000010000000391291de76ce6e24d1e2a
9b60266519bc8ce889f814deb0fc00edd3129de3ab9b6bfa3bf47d007d844af7db74
9ea97215e82f456cbdd473812c6a042ae39539898752c89b60a276ec8a9feab900e2
5bdfe0ab8e773aa1c36ae214d67c65bb68630450a5db2c7c6403b77f6a9bf4d30a02
19db5cced884d7514f3cbd19220020bf3045b0e5c6955b32864f16f97da02f0cbfea
70458b07032e30b0342d75b8f3dc6871442e6384b10f559f5dc594a214924c48ccc3
37078665653fc740340428138b0fb5154f2f2cb291ad05ace7acae60031b2d09b2f4
17712d1c01e34b165af2e070f5a521a85a5fb3dd2a6288947bcbd5e2265d3670bd61
92eb2bf643964e2783d84aec343f8e3571e4fcf09cbeea94e80470aa7252d1c733a5
535907e66c7b9f0b88b159dc2a7370ee47f13e7e134d3d05e5f53fac640b784a9b0f
183fe14217325626f487cc8d8cb9eaf0abb174ee0b7076cf39c45037cefdf3f1e61b
5174581214c09870b72c39737ec4c46a96199b66cad2990bcbe5bb1abfde99107c7f
7289395bf2a433598ede0b1969f23db949afb5b4d33831dae6c641a6355f8f9bf16c
dffc4bf86891b93a557c2152ac8a1de51c995344cc10cc4bc9ecfbb4e418bed0f334
af165339e6725dc4fc1e995521e1be8a566d59b57cd130903b42d07087d63646ef8f
c1e9e9071bb67a123fdec3f37638cdaf0f4bf3084074069171c17885b9431ad908d3
6a6f8a826256d2aa34f8aa0731a357c060db8e80fefd61b1c323890e640633b98d17
5d4d6ebff800a71cfc864ec02837de9d0e079f0f400acafd56805cb273e631ba395d
23e86acf6eae63181a5afe1f0a361cbbd5fefeb7db0c95591ec3128e80dfbea9ca0f
89fc035d761c05d41e7a010892c42e8e2af62aa604f4e214c0bb08075481f9cc307a
555adf333b9424f209b89f161032e413b047ae5ab0aa15643bb4c643446d2c9829eb
256e7375ce9639047a24a44f4da446b7359556f3ab3484c56511c68a140dc0531f65
3105800d9f20990d4ebdc5ceea918d7ae95c0d7ec69a00d6a936b25fc19b9dfc5561
400f046191136c367038d6a9d0e0ae30dcdc4733712cbd5a2aee35315eff5c1a7e08
5b68c5cf0c64c495df2ca6f030db04480a2e11d4a0a0dbf29d9463d5b9e41e346e49
c894d5e43993c834c4746309c886d6131f2f92155ca1160bac9660802a947b5aba94
b35357d13fdf02d2aeabef568912f68ae5d3a60214f6d00c4dd9f0af09eb0bf961cd
9f27251d46899c28d87080ba2ead3e8193f51a789706ec32aacee9f4b14eeca91a25
2fe894b30dc3938abbbe7d217948cae79ce3adb4d7d7df6756f3099f2543ed3b522b
acab257503c9e07fcd32cc32fa9aa17977ec05bc5fe0f5954d51f160f52d33f93166
af68aa90261b3f5ad273adacf2d0cb5b0c5402bfa62da67a52dcddfa463e72d2c005
f1ac0ea3cb62364ee3419333612e07bf685006137a592e2fcd58398265c4ff9e11e7
0c2b79152e4604b4f94676e955bcff4dfc429a8a88728b95bfc2826e25ba6eab9cfb
066c9911693efff242f7b51c3cb88546143b8ab2142dd3c9bda55d16fe3084a86b74
3f294dd9d0aa84f3ce3b083a5879a4762a756e9b41f4bdf8b71418073b0a0d4a9c13
1882455ece23e50324c5feea217920b0f3109dcbdc81762e41b7ca271efac8e39cc2
6ebe085abdbf6b314a38929799fb7feebee2e20b97056ed17ef3881e6e89330314dd
7e9c629c46dfdb925c7c5f5d243f159d964691745cd46579fd0696479e1c49cbd2af
879a2bce8576619cca7b6e516e6c94c1087441a81f11b9a83535b24ddc725a81a9d1
ff62da2804c8d84c6e382065574282ea1f23eaf648cfa9767afb098fd81654d76133
f5f39bcc762c9bc31f7f4665cc0efa929b5c05dedd76143c63dc7018ab130c108ea9
01be32b9d911b66da13a1528c32a9694c899a772f8e1fe00c17eceb343e737d72cba
06cf5ddac9a4d3df7ef391cf6595a6d8c14b0d80f93023b1b3d4371239da98b67a1b
6a379422616282a16e8d1f97a130baf21e572bcca91abb760eac6957f9b1b05e49e2
d181874ac6dd160d1c717b73bd28ef55f08d47466d5aef754814c7e206fa9e2ec533
85d14d52f7769d95ea50524ffb20dc7275b04d71d1967e3bbc6ed481f1fc5a15e78a
1fd967d96045625645dbd173cccdd97661e995ce47d6b3ead96ee6d006a5ce6f4c97
77fe2e3f91bebe877cac8c6486dfce0315dc71bbb93879759b8981c5ff2e11deb809
abf4280ee93d1711e73645b410acb518538ce3d4bda1e355c988f068165668e99d6a
8de356b4b13298036ad05d526c4a5e2591612a477b7e86550adde128cd71ee651d44
99699000a02979e42bbccf32c83b1eb0ff99aa4d352e20e0b3382422df2c2ed4ce90
c94cf1a359e92ef971dc6db06047a333c2ebe827eb6d5f2811fdbe0bf0f12bf2094e
0dcd8e418f3f691a60ceb0cefb6f45f47883d6b9f320950e91266740c6dbfad6b3cf
e56de0aa6658b0dc893bb6e49e6294537a7878e86cfc8e6c150675db1a89d188ea6e
fe7d88ff57b39b8610e392811ee097ca61c4841e0fbd346ed3ff6a5e412acb0d9f13
022df2e7fdaa8e0face7366c8ffe6f446995b564fc3d59c70fecdb60a25e28650417
```

```
157f43f3e72c3afc601509641cfd099a78130e1f7ba8333502ad4f036f46411a43d0
35e2ca0ed0c346d9aac5df05196c95c38e6e52763ed896b6d02464a910dda6cca340
24e3b9c3723d26e2886ad724dd56ea285e8e4b60beec924d55dd700c38877b74552f
ea1f8741579b02061416131db390f628522885236b51f7aef23167d3a5fe5eadcd88
b0e99b2b6bc56b0dea4fb22146294766c28e5e7c834dbdcb6bfdd7bd8455252522ff
2e974f6fd3fda176749b7cdced5b9aba092b2982c89cb7d2b36348928c8f01170618
ecff14d9e0eed9d88d97e38bcf7a837f674be5243fc624c8afd3d105f462bfa939b8
143a3a98f78fbb8c915e00bdbbf707b12c45784f4d1cb1426b583a0d5fbec1f5ea6d
0067c090168cb788e532aca770c7be366ec07e7808f1892b00000006ed1ce8c6e437
918d43fba7bd9385694c41182703f6b7f704deedd9384ba6f8bc362c948646b3c984
8803e6d9ba1f7d3967f709cddd35dc77d60356f0c36808900b491cb4ecbbabec128e
7c81a46e62a67b57640a0a78be1cbf7dd9d419a10cd8686d16621a80816bfdb5bdc5
6211d72ca70b81f1117d129529a7570cf79cf52a7028a48538ecdd3b38d3d5d62d26
246595c4fb73a525a5ed2c30524ebb1d8cc82e0c19bc4977c6898ff95fd3d310b0ba
e71696cef93c6a552456bf96e9d075e383bb7543c675842bafbfc7cdb88483b3276c
29d4f0a341c2d406e40d4653b7e4d045851acf6a0a0ea9c710b805cced4635ee8c10
7362f0fc8d80c14d0ac49c516703d26d14752f34c1c0d2c4247581c18c2cf4de48e9
ce949be7c888e9caebe4a415e291fd107d21dc1f084b1158208249f28f4f7c7e931b
a7b3bd0d824a4570'
      ]
    ]
  ]
)
```

A.2. Example COSE_Sign1 Message

This section provides an example of a COSE_Sign1 message.

The 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'00000000000000000000000391291de76ce6e24d1e2a9b60
266519bc8ce889f814deb0fc00edd3129de3ab9b9aa5b5ac783bdf0fe689f57fb204
f1992dbc1ce2484f316c74bce3f2094cfa8e96a4a9548cead0f78ee5d549510d1910
f647320448ae27ecce77249802a0c39c645bf8db08573af52c93d91fd0e217f245c7
52c176b81514eb6e3067e0fbb329225eaa88c7d21635e32ae84213f89018cb06f1b8
4e61eac348b690d7c6265c19f9d868952d99826aecd417b5279dd674cd951c306016
cfee4fee3bfcf5ee5a5ad08b5b4f53bc93995f26cfe7c0c1c5ba2574c1f2d8470993
e8bd47ef9b9cf309ef895226e92be60683459009611defbb9a43217956a0ab2959bb
da0feca39de37e7c4a6cd8a5314d6b02b377406d5a5e589e91feaa9f2e4ec1682ba1
f633c7784499323e40da651f71d3c19e38c634d898b0c508324c0bfcf7c5f0a8c014
b4af200a739f96cddba94daf86ce80c76158d4f5cf3cd2ba9f1393df47e556887f91
68540485242a05ec6bcc76659ec3d0d2fedae3fd1608a701c226f5fd83c9b1ed3152
ddac7426c30e3390bec8f1da6174abe8d3568c9b76b149eb077d61ac15b8fb11b8ce
5f9d14e448e216f375e1f96a52d39619459b131026143e8809bad408f5ef66cd3da2
27431e68670c0b4b2c3801e1e9025b1ebed218e0956967158ccc274c704adcd8cc23
c149a89eda25478742dadc15f233844535e4021000b5d557313d4f271875680e6d5e
7f6681fdd19f8b9a748cabb2377aac1387fdb80e618eb7d69a368729ca9a092af91e
be1c584c35fe62734d1d53d10b35dd02093a201c889ad37a558b610f1ab00179a11f
881600e944cedc47a7ae6d828009d7c61ffea9dd5aa5406408e2e85dc056e47b5758
9eaba18e792f4631af62d4588a1818167274273c69e7a0735be5dada7e224e3b178b
3b093212eb74e762f564a26d577aa22ebd8c7b4a999419908e2f2d9c8689dc923905
c198b9ee335d1e0de6d689655f446dffea997b6e58f5f648415233ede3b9d8a2db29
e8c3dde5d8dbd55e6348cd9f421783db090e087de46425d62d513597b00d7de32fad
87752a79cee8b2a38b1e0f2562836721cbbfba20f131130c009a436b93a0bb44fcbb
86228b1bf1a35f4fc626817924eaebd5b78d64a7970d18dade90cf0ad759b1c45d95
3c08cd1189685077c5a56069da0944669d797496f8f886fea6f792598db2ac66b657
af838ed3c3a914dffbb164170a1f63250b125eda53ecaeaf6ee0d2b8a3c804104d7e
d575b66469bc59f37eec6c6f6fb19e0f7ea02d7c85306230063adb58950589f6ffaf
f1407233828ae0dfbe5889e5de00bb640a4bc24c3f704488fa669676a9ebbbed399b
8a9ac0ee4cc944f864b21f642e04f610319ac9271f8bd820e77e41dac6553d234d94
80e26142c0fa37416651d6450e1f2082bd0213d6783e1ae3cc5c5af677c3316e173b
a4716d6bc8a9d89383f8b025a0859b99a43daeaf8ddaed46d223b9b503651a67560b
```

```
   feb2f35ba544722620ec4086dcc77e6e87bb53f1f18c38368662be460ede31325cae
   aebf018a6fa9d32e3c3a6898e15fe114dcce51241c61afabc36de3608b4d342712a8
   33615c6131e89e1d46b713d9638a08b5a768d53af0298b9c874ded7084358223840c
   2e78cd6fbfca695279a4c1883bb7de81b04a069de8277f7f5109c16938347a643713
   c9ac36fffc8bf141e899f48bc25c7b636d43bebcfa7742d4e1462263e56732ad2021
   eef8ce84023c4959cfd250343d62074724907de9d49ea2f6c968fd9e9bf28feafcdc
   81702108805dec60f2781272d2425a6ee29c66122d2c557867c1a5aed82131e06fc3
   84ecf49017e1c9d6cf63b9f2285ccf890cbb9bbf796e0fd02101948b7ef663849367
   7b33fd787d9d3fc2c7cc7babc21af8c748afb80cf86b45dc89f0b9c7959621e85b98
   b542dc263db9255273bb9054a7f194748f28373ba123d73fc71fef43e7e2ac9a8000
   8e85cf2f04aa433075dfc54c4de24a341ebf7cf1e6b383dbba85898fdc368017fd67
   c153e7a991a3a3cee6dae4fbe2fe6f25a8df314140a8176c8e6fd0c6f042ca66eb6a
   bba9a2502bb6dfa52960ae86a942a673e4e45439594fefcd2974e20554d1dc70b8e0
   34fd1787801343d5f6edc95ce0348c25727c771526e3fd4effb5f16e25a1ea3dcd82
   82e778e91ae9b339a5013c77fd6ea2432704e293f5e82a24121c73900bea4b4ef14a
   2adc1ab3c68224bae1de9c61a48b84e84c1b0e83701be3d988012a24fa40268c8d6e
   f1fd2818ae8e4b6f52f89beab6bfdd1ff1b7ecd573edff3703b800b5b2a206f451f1
   bf2713b4ae9085bd7fe34ad4306a290e4cdb7817ee9ab7ccfb816d002b619f77d46d
   7dd0f8eefe10f5c0f9723ffdb14ca75a185543770f41508b9983d5eed78225bc6e21
   f876bfdd08fe8bc63e0cb253c7dfc67c330897c515244f3f631682f2141eba48ca86
   dfff9206f78edcb9dec4b2371aeddbe141ef96a10957e29a94747c4438fb30b14d37
   e7428eb7fbe4f9d870e72f35f55847f230374bdf56dcae6c129b4468ebaedc340ff4
   cc160c6b410e2d8989488ac8ef9a9febbf65ad4fdfba532a8122ef82dc1a4ffc361c
   bf9f752b36aa9821683d5f3f5842f90134eb423d5cbc76858b4c0a7ba798ec94a089
   fdb24b5b25f42d7b6bb8192f07b98eb2de1fe7bc8b6c740fa5cde6fb4890d2f17916
   64a96c25a0a71a541025b5ec825eed91f393505473e21d0620177993982e6c1b6bf9
   1b777b5ab5739b84946c518c7e6aa0e689e9ad1d34e6ef6ca0e709c4aefecd6f2594
   b017940742aceb72c5a52d7d47a3a74f9d09eb84cf82b349de32278a771cebc31ebc
   580c09b11799b1f0e6d11d75b17e389d259c531f957a1e699250711df2e36f64f21c
   92eff698a392d92df0b2f91991408a076b83149e025a9ffba1ff1caed916a2fc1ac5
   d3081c30b5c64b7d677c314b6e76ac20ed8bb4a4c0eb465ae5c0c265969264b27e6d
   54c266f79e58e2fa6a381069090bec00189562abcf831adc86a05a2fc7ffaa70dbd3
   fa60e09d447cd76b2ff2b851c38e72650ade093ba8bd000000067b95de445abf8916
   1dff4b91a4a9e3bf156a39a4660f98f06bf3f017686d9dfc362c948646b3c9848803
   e6d9ba1f7d3967f709cddd35dc77d60356f0c36808900b491cb4ecbbabec128e7c81
   a46e62a67b57640a0a78be1cbf7dd9d419a10cd8686d16621a80816bfdb5bdc56211
   d72ca70b81f1117d129529a7570cf79cf52a7028a48538ecdd3b38d3d5d62d262465
   95c4fb73a525a5ed2c30524ebb1d8cc82e0c19bc4977c6898ff95fd3d310b0bae716
   96cef93c6a552456bf96e9d075e383bb7543c675842bafbfc7cdb88483b3276c29d4
   f0a341c2d406e40d4653b7e4d045851acf6a0a0ea9c710b805cced4635ee8c107362
   f0fc8d80c14d0ac49c516703d26d14752f34c1c0d2c4247581c18c2cf4de48e9ce94
   9be7c888e9caebe4a415e291fd107d21dc1f084b1158208249f28f4f7c7e931ba7b3
   bd0d824a4570'
     ]
   )
```

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