Workgroup:	lamps
RFC:	9708
Obsoletes:	8708 (if approved)
Published:	December 2024
Intended Status:	Standards Track
Expires:	23 June 2025
Author:	R. Housley
	Vigil Security

Use of the HSS/LMS Hash-Based Signature Algorithm in the Cryptographic Message Syntax (CMS)

Abstract

This document specifies the conventions for using the Hierarchical Signature System (HSS) / Leighton-Micali Signature (LMS) hash-based signature algorithm with the Cryptographic Message Syntax (CMS). In addition, the algorithm identifier and public key syntax are provided. The HSS/LMS algorithm is one form of hash-based digital signature; it is described in RFC 8554. This document obsoletes RFC 8708.

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 4 June 2025.

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 Cryptographic Message Syntax (CMS) [CMS] signed-data content type. 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 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 with a given private key, and 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 are defined in [HASHSIG]. Currently, parameter sets are defined that use SHA-256 [SHS] and SHAKE256 [SHA3].

1.1. ASN.1

CMS values are generated using ASN.1 [ASN1-B], using the Basic Encoding Rules (BER) and the Distinguished Encoding Rules (DER) [ASN1-E].

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.

1.3. Motivation

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

If cryptographically relevant quantum computers (CRQCs) are ever built, 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 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, Elliptic Curve Digital Signature Algorithm (ECDSA), and Edwards-curve Digital Signature Algorithm (EdDSA) are all vulnerable if CRQCs are ever developed.

Since the HSS/LMS signature algorithm does not depend on the difficulty of discrete logarithms or factoring, but on a second-preimage-resistant cryptographic hash function, the HSS/LMS signature algorithm is considered to be post-quantum secure. One use of post-quantum-secure signatures is the protection of software updates, perhaps using the format described in [FWPROT], to enable deployment of software that implements new cryptosystems.

1.4. Changes Since RFC 8708

At the time RFC 8708 was published, there were no plans to put an HSS/LMS public key in a certificate. The expectation was that the HSS/LMS public key would be distributed by some other means. Today, there are plans to put an HSS/LMS public key in a certificate [X.509-S-HBS]. The

KEY field of the pk-HSS-LMS-HashSig definition in the ASN.1 module does not come into play when using HSS/LMS signatures in the CMS; however, it needs to be consistent with the conventions for carrying an HSS/LMS public key in a certificate. The pk-HSS-LMS-HashSig definition is updated to reflect no ASN.1 wrapping for the public key. These changes resolve [Err7960] and [Err7963].

Additional HSS/LMS tree sizes have been defined. The list in Section 2.2 was expanded to include the recently defined ones.

Additional LM-OTS Signatures have been defined. The list in Section 2.3 was expanded to include the recently defined ones.

More detail has been provided in Section 4 regarding allowed values in the X.509 certificate key usage extension for an HSS/LMS public key.

2. HSS/LMS Hash-Based Signature Algorithm Overview

Merkle Tree Signatures (MTS) are a method for signing a large but fixed number of messages. An MTS system depends on a one-time signature method and a collision-resistant hash function.

This specification makes use of the hash-based 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].

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] uses only the SHA-256 one-way hash function [SHS], but it establishes an IANA registry [IANA-LMS] to permit the registration of additional one-way hash functions in the future.

2.1. Hierarchical Signature System (HSS)

The MTS system 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 standalone tree (a top tree with no children) can be summarized as:

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

where, u32str() and || are used as defined in [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 */
```

where, as defined in Section 3.3 of [HASHSIG], the signed_public_key structure contains 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 system 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] specification supports five values for this parameter: h=5, h=10, h=15, h=20, and h=25. There are 2^h leaves in the tree. The second parameter, m, is the number of bytes output by the hash function, and it is the amount of data associated with each node in the tree. The [HASHSIG] specification supports the SHA-256 hash function [SHS], with m=32. Additional LMS Signature parameter sets have been registered at [IANA-LMS].

As specified in [HASHSIG], the LMS public key consists of four elements: the lms_algorithm_type from the list above, the otstype to identify the Leighton-Micali One-Time Signature (LM-OTS) type as discussed in Section 2.3, the private key identifier (I) as described in Section 5.3 of [HASHSIG], and the m-byte string associated with the root node of the tree (T[1]).

The LMS public key can be summarized as:

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

As specified in [HASHSIG], an LMS signature consists of four elements: the number of the leaf (q) associated with the LM-OTS signature value, an LM-OTS signature value 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 value 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

Merkle Tree Signatures (MTS) depend on a one-time signature method, and [HASHSIG] specifies the use of the LM-OTS, which has five parameters:

n: The length in bytes of the hash function output.

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 value.

ls: The number of bits that are left-shifted in the final step of 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] specifies four LM-OTS variants. Additional LM-OTS Signature parameter sets have been registered at [IANA-LMS].

Signing involves the generation of C, 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. Algorithm Identifiers and Parameters

The algorithm identifier for an HSS/LMS hash-based signature is:

```
id-alg-hss-lms-hashsig OBJECT IDENTIFIER ::= { iso(1)
    member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
    smime(16) alg(3) 17 }
```

When this object identifier is used for an HSS/LMS signature, the AlgorithmIdentifier parameters field **MUST** be absent (that is, the parameters are not present, and the parameters are not set to NULL).

In the CMS, the HSS/LMS signature value is a large OCTET STRING. The HSS/LMS signature generation is described in Section 2 of this document. The signature format is designed for easy parsing. The HSS, LMS, and LM-OTS components of the signature value each include a counter and a typecode that indirectly provide all of the information that is needed to parse the value during signature validation.

The signature value identifies the hash function used in the HSS/LMS tree.

4. HSS/LMS Public Key Identifier

The AlgorithmIdentifier for an HSS/LMS public key uses the id-alg-hss-lms-hashsig object identifier, and the parameters field **MUST** be absent.

When this AlgorithmIdentifier appears in the SubjectPublicKeyInfo field of a certification authority (CA) X.509 certificate [RFC5280], the certificate key usage extension **MUST** contain at least one of the following values: digitalSignature, nonRepudiation, keyCertSign, and cRLSign. However, it **MUST NOT** contain other values.

When this AlgorithmIdentifier appears in the SubjectPublicKeyInfo field of an end-entity X.509 certificate [RFC5280], the certificate key usage extension **MUST** contain at least one of the following: digitalSignature, nonRepudiation, or cRLSign. However, it **MUST NOT** contain other values.

```
pk-HSS-LMS-HashSig PUBLIC-KEY ::= {
    IDENTIFIER id-alg-hss-lms-hashsig
    -- KEY no ASN.1 wrapping --
    PARAMS ARE absent
    CERT-KEY-USAGE
      { digitalSignature, nonRepudiation, keyCertSign, cRLSign } }

HSS-LMS-HashSig-PublicKey ::= OCTET STRING
```

The id-alg-hss-lms-hashsig algorithm identifier is also referred to as id-alg-mts-hashsig. This synonym is based on the terminology used in an early draft version of the document that became [HASHSIG].

When the public key appears outside a certificate, it is an OCTET STRING. Like the signature format, it is designed for easy parsing. The value is the number of levels in the public key, L, followed by the LMS public key.

The HSS/LMS public key value can be described as:

```
      u32str(L) || lms_public_key
```

The public key for the topmost LMS tree is the public key of the HSS system. When L=1, the HSS system is a single tree.

5. Signed-Data Conventions

As specified in [CMS], the digital signature is produced from the message digest and the signer's private key. The signature is computed over different values depending on whether signed attributes are absent or present.

When signed attributes are absent, the HSS/LMS signature is computed over the content. When signed attributes are present, a hash is computed over the content using the same hash function that is used in the HSS/LMS tree, then a message-digest attribute is constructed with the hash of the content, and then the HSS/LMS signature is computed over the DER-encoded set of signed attributes (which **MUST** include a content-type attribute and a message-digest attribute). In summary:

```
IF (signed attributes are absent)
THEN HSS_LMS_Sign(content)
ELSE message-digest attribute = Hash(content);
     HSS_LMS_Sign(DER(SignedAttributes))
```

When using [HASHSIG], the fields in the SignerInfo are used as follows:

- digestAlgorithm **MUST** contain the one-way hash function used in the HSS/LMS tree. For convenience, the AlgorithmIdentifier for SHA-256 from [PKIXASN1] and the AlgorithmIdentifier for SHAKE256 from [RFC8692] are repeated here:

```
mda-sha256 DIGEST-ALGORITHM ::= {
    IDENTIFIER id-sha256
    PARAMS TYPE NULL ARE preferredAbsent }

id-sha256 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
    country(16) us(840) organization(1) gov(101) csor(3)
    nistAlgorithms(4) hashalgs(2) 1 }

mda-shake256 DIGEST-ALGORITHM ::= {
    IDENTIFIER id-shake256 }

id-shake256 OBJECT IDENTIFIER ::= { joint-iso-itu-t(2)
    country(16) us(840) organization(1) gov(101) csor(3)
    nistAlgorithm(4) hashAlgs(2) 12 }
```

- signatureAlgorithm **MUST** contain id-alg-hss-lms-hashsig, and the algorithm parameters field **MUST** be absent.
- signature contains the single HSS/LMS signature value resulting from the signing operation as specified in [HASHSIG].

6. Security Considerations

Implementations **MUST** protect the private keys. Compromise of the private keys will 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 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 pseudorandom number generator (PRNG) to generate these values is much less severe than in the generation of private keys, the guidance in [RFC4086] remains important.

When computing signatures, the same hash function **SHOULD** be used to compute the message digest of the content and the signed attributes, if they are present.

7. IANA Considerations

In the "SMI Security for S/MIME Module Identifier (1.2.840.113549.1.9.16.0)" registry, IANA has changed the reference for value 64 to this document.

In the "SMI Security for S/MIME Algorithms (1.2.840.113549.1.9.16.3)" registry, IANA has changed the reference for value 17 to this document.

8. References

8.1. Normative References

[ASN1-B] ITU-T, "Information technology -- Abstract Syntax Notation One (ASN.1): Specification of basic notation", ITU-T Recommendation X.680, August 2015.

[ASN1-E] ITU-T, "Information technology -- ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, August 2015.

[CMS] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/info/rfc5652>.

[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>.

[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>.

[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>.

[RFC8692] Kampanakis, P. and Q. Dang, "Internet X.509 Public Key Infrastructure: Additional Algorithm Identifiers for RSASSA-PSS and ECDSA Using SHAKEs", RFC 8692, DOI 10.17487/RFC8692, December 2019, <https://www.rfc-editor.org/info/rfc8692>.

[SHA3] National Institute of Standards and Technology, "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions", DOI 10.6028/NIST.FIPS.202, FIPS PUB 202, August 2015, <https://doi.org/10.6028/NIST.FIPS.202>.

[SHS] National Institute of Standards and Technology, "Secure Hash Standard (SHS)", FIPS PUB 180-4, DOI 10.6028/NIST.FIPS.180-4, August 2015, <https://doi.org/10.6028/NIST.FIPS.180-4>.

8.2. 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>.

[CMSASN1] Hoffman, P. and J. Schaad, "New ASN.1 Modules for Cryptographic Message Syntax (CMS) and S/MIME", RFC 5911, DOI 10.17487/RFC5911, June 2010, <https://www.rfc-editor.org/info/rfc5911>.

[CMSASN1U] Schaad, J. and S. Turner, "Additional New ASN.1 Modules for the Cryptographic Message Syntax (CMS) and the Public Key Infrastructure Using X.509 (PKIX)", RFC 6268, DOI 10.17487/RFC6268, July 2011, <https://www.rfc-editor.org/info/rfc6268>.

[Err7960] RFC Errata, Erratum ID 7960, RFC 8708, <https://www.rfc-editor.org/errata/eid7960>.

[Err7963] RFC Errata, Erratum ID 7963, RFC 8708, <https://www.rfc-editor.org/errata/eid7963>.

[FWPROT] Housley, R., "Using Cryptographic Message Syntax (CMS) to Protect Firmware Packages", RFC 4108, DOI 10.17487/RFC4108, August 2005, <https://www.rfc-editor.org/info/rfc4108>.

[IANA-LMS] IANA, "Leighton-Micali Signatures (LMS)", <https://www.iana.org/assignments/leighton-micali-signatures/>.

[LM] Leighton, T. and S. Micali, "Large provably fast and secure digital signature schemes based on 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, 1979.

[M1987] Merkle, R., "A Digital Signature Based on a Conventional Encryption Function", Advances in Cryptology -- CRYPTO '87 Proceedings, Lecture Notes in Computer Science, Vol. 293, 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, Vol. 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, Vol. 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, <https://doi.org/10.17226/25196>.

[PKIXASN1] Hoffman, P. and J. Schaad, "New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)", RFC 5912, DOI 10.17487/RFC5912, June 2010, <https://www.rfc-editor.org/info/rfc5912>.

[PQC] Bernstein, D., "Introduction to post-quantum cryptography", Post-Quantum Cryptography, Springer, pp. 1-14, DOI 10.1007/978-3-540-88702-7_1, 2009, <https://doi.org/10.1007/978-3-540-88702-7_1>.

[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>.

[X.509-S-HBS] Van Geest, D., Bashiri, K., Fluhrer, S., Gazdag, S., and S. Kousidis, "Use of the HSS and XMSS Hash-Based Signature Algorithms in Internet X.509 Public Key Infrastructure", Work in Progress, Internet-Draft, draft-ietf-lamps-x509-shbs-13, 12 December 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-x509-shbs-13>.

Appendix A. ASN.1 Module

The ASN.1 module in this appendix builds upon the modules in [CMSASN1] and [CMSASN1U].

```
<CODE BEGINS>
MTS-HashSig-2013
  { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
    id-smime(16) id-mod(0) id-mod-mts-hashsig-2013(64) }

DEFINITIONS IMPLICIT TAGS ::= BEGIN

EXPORTS ALL;

IMPORTS
  PUBLIC-KEY, SIGNATURE-ALGORITHM, SMIME-CAPS
    FROM AlgorithmInformation-2009  -- RFC 5911 [CMSASN1]
      { iso(1) identified-organization(3) dod(6) internet(1)
        security(5) mechanisms(5) pkix(7) id-mod(0)
        id-mod-algorithmInformation-02(58) } ;

--
-- Object Identifiers
--

id-alg-hss-lms-hashsig OBJECT IDENTIFIER ::= { iso(1)
    member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
    smime(16) alg(3) 17 }

id-alg-mts-hashsig OBJECT IDENTIFIER ::= id-alg-hss-lms-hashsig

--
-- Signature Algorithm and Public Key
--

sa-HSS-LMS-HashSig SIGNATURE-ALGORITHM ::= {
    IDENTIFIER id-alg-hss-lms-hashsig
    PARAMS ARE absent
    PUBLIC-KEYS { pk-HSS-LMS-HashSig }
    SMIME-CAPS { IDENTIFIED BY id-alg-hss-lms-hashsig } }

pk-HSS-LMS-HashSig PUBLIC-KEY ::= {
    IDENTIFIER id-alg-hss-lms-hashsig
    -- KEY no ASN.1 wrapping --
    PARAMS ARE absent
```

```
    CERT-KEY-USAGE
        { digitalSignature, nonRepudiation, keyCertSign, cRLSign } }

HSS-LMS-HashSig-PublicKey ::= OCTET STRING

--
-- Expand the signature algorithm set used by CMS [CMSASN1U]
--

SignatureAlgorithmSet SIGNATURE-ALGORITHM ::=
    { sa-HSS-LMS-HashSig, ... }

--
-- Expand the S/MIME capabilities set used by CMS [CMSASN1]
--

SMimeCaps SMIME-CAPS ::=
    { sa-HSS-LMS-HashSig.&smimeCaps, ... }

END

<CODE ENDS>
```

Acknowledgements

Many thanks to the people that provided comments on the draft documents that resulted in RFC 8708. Thanks to Joe Clarke, Roman Danyliw, Scott Fluhrer, Jonathan Hammell, Ben Kaduk, Panos Kampanakis, Barry Leiba, John Mattsson, Jim Schaad, Sean Turner, Daniel Van Geest, and Dale Worley for their careful review and comments.

Many thanks to Daniel Van Geest for motivating the creation of this document.

Author's Address

Russ Housley
Vigil Security, LLC
516 Dranesville Road
Herndon, VA 20170
United States of America
Email: housley@vigilsec.com