Stream: Internet Research Task Force (IRTF)
RFC: 9858
Category: Informational
Published: October 2025
ISSN: 2070-1721
Authors: S. Fluhrer, *Cisco Systems*; Q. Dang, *NIST*

RFC 9858
Additional Parameter Sets for HSS/LMS Hash-Based Signatures

Abstract

This document extends HSS/LMS (RFC 8554) by defining parameter sets that use alternative hash functions. These include hash functions that result in signatures with significantly smaller sizes than the signatures that use the RFC 8554 parameter sets and should have sufficient security.

This document is a product of the Internet Research Task Force (IRTF). The IRTF publishes the results of Internet-related research and development activities. These results might not be suitable for deployment. This RFC represents the consensus of the Crypto Forum Research Group of the Internet Research Task Force (IRTF). Documents approved for publication by the IRSG are not candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Research Task Force (IRTF). The IRTF publishes the results of Internet-related research and development activities. These results might not be suitable for deployment. This RFC represents the consensus of the Crypto Forum Research Group of the Internet Research Task Force (IRTF). Documents approved for publication by the IRSG are not candidates for any level of Internet Standard; see 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/rfc9858.

Copyright Notice

Copyright (c) 2025 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.

Table of Contents

1. Introduction

Stateful hash-based signatures have small private and public keys, are efficient to compute, and are believed to have excellent security. One disadvantage is that the signatures they produce tend to be somewhat large (possibly 1-4 kilobytes). This document defines a set of parameter sets for the HSS/LMS stateful hash-based signature method [RFC8554] that reduce the size of the signature significantly or rely on a hash function other than SHA-256 (to increase cryptodiversity).

This document represents the consensus of the Crypto Forum Research Group (CFRG) in the IRTF. It is not an IETF product and is not a standard.

According to official definitions and common usage, a Leighton-Micali Signature (LMS) is a stateful hash-based signature scheme that is based on a single-level Merkle tree. The Hierarchical Signature System (HSS) is a way of binding several LMS signatures together in a hierarchical manner to increase the number of signatures available. Strictly speaking, all the signatures discussed in this document are HSS signatures (even if the HSS signature consists of a single LMS signature). However, it is common to refer to these signatures as "LMS signatures". This document uses the term "HSS/LMS" to cover both the pedantic and the common meanings.

This document is intended to be compatible with the NIST document [NIST_SP_800-208].

2. Additional Hash Function Definitions

This section defines three hash functions that are used with the parameter sets defined in Sections 3 and 4. These hash functions are used where SHA-256 is used in the original parameter sets from [RFC8554]. The hash function used is specified by the parameter set that is selected.

2.1. 192-Bit Hash Function Based on SHA-256

This document defines a SHA-2-based hash function with a 192-bit output. As such, we define SHA-256/192 as a truncated version of SHA-256 [FIPS180]. That is, it is the result of performing a SHA-256 operation to a message and then omitting the final 64 bits of the output. This procedure for truncating the hash output to 192 bits is described in Section 7 of [FIPS180].

The following test vector illustrates this:

```
SHA-256("abc")     = ba7816bf 8f01cfea 414140de 5dae2223
                     b00361a3 96177a9c b410ff61 f20015ad
SHA-256/192("abc") = ba7816bf 8f01cfea 414140de 5dae2223
                     b00361a3 96177a9c
```

We use the same initial hash value (initialization vector) as the untruncated SHA-256, rather than defining a distinct one, so that we can use a standard SHA-256 hash implementation without modification. In addition, the fact that anyone gets partial knowledge of the SHA-256 hash of a message by examining the SHA-256/192 hash of the same message is not a concern for this application. Each message that is hashed is randomized. Any message being signed includes the C randomizer (a value that is selected by the signer and is included in the hash), which varies per message. Therefore, signing the same message by SHA-256 and by SHA-256/192 will not result in the same value being hashed, and so the latter hash value is not a prefix of the former one. In addition, all hashes include the I identifier, which is included as a part of the signature process in [RFC8554]. This I identifier is selected randomly for each private key (and hence two keys will have different I values with high probability), and so two intermediate hashes computed as a part of signing with two HSS private keys (one with a SHA-256 parameter set and one with a SHA-256/192 parameter set) will also be distinct with high probability.

2.2. 256-Bit Hash Function Based on SHAKE256

This document defines a SHAKE-based hash function with a 256-bit output. As such, we define SHAKE256/256 to be the first 256 bits of the SHAKE256 extendable-output function (XOF). That is, it is the result of performing a SHAKE-256 operation to a message and then using the first 256 bits of output. See [FIPS202] for more detail.

2.3. 192-Bit Hash Function Based on SHAKE256

This document defines a SHAKE-based hash function with a 192-bit output. As such, we define SHAKE256/192 to be the first 192 bits of the SHAKE256 XOF. That is, it is the result of performing a SHAKE-256 operation to a message and then using the first 192 bits of output. See [FIPS202] for more detail.

3. Additional LM-OTS Parameter Sets

The table below defines the Leighton-Micali One-Time Signature (LM-OTS) parameters that use the hashes defined in Section 2:

Parameter Set Name	H	n	w	p	ls	id
LMOTS_SHA256_N24_W1	SHA-256/192	24	1	200	8	0x00000005
LMOTS_SHA256_N24_W2	SHA-256/192	24	2	101	6	0x00000006
LMOTS_SHA256_N24_W4	SHA-256/192	24	4	51	4	0x00000007
LMOTS_SHA256_N24_W8	SHA-256/192	24	8	26	0	0x00000008
LMOTS_SHAKE_N32_W1	SHAKE256/256	32	1	265	7	0x00000009
LMOTS_SHAKE_N32_W2	SHAKE256/256	32	2	133	6	0x0000000A
LMOTS_SHAKE_N32_W4	SHAKE256/256	32	4	67	4	0x0000000B
LMOTS_SHAKE_N32_W8	SHAKE256/256	32	8	34	0	0x0000000C

Parameter Set Name	H	n	w	p	ls	id
LMOTS_SHAKE_N24_W1	SHAKE256/192	24	1	200	8	0x0000000D
LMOTS_SHAKE_N24_W2	SHAKE256/192	24	2	101	6	0x0000000E
LMOTS_SHAKE_N24_W4	SHAKE256/192	24	4	51	4	0x0000000F
LMOTS_SHAKE_N24_W8	SHAKE256/192	24	8	26	0	0x00000010

Table 1

Parameter Set Name: The human-readable name of the parameter set.

H: The second-preimage-resistant cryptographic hash function used within this parameter set.

n: The number of bytes of the output of the hash function.

w: The width (in bits) of the Winternitz coefficients; that is, the number of bits from the hash or checksum that is used with a single Winternitz chain. It is a member of the set { 1, 2, 4, 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 Cksm (used by algorithm 2 of [RFC8554]).

id: The IANA-defined identifier used to denote this specific parameter set, which appears in both public keys and signatures.

These values are additions to the entries in Table 1 of [RFC8554].

The SHA256_N24, SHAKE_N32, and SHAKE_N24 in the parameter set names denote the SHA-256/192, SHAKE256/256, and SHAKE256/192 hash functions defined in Section 2.

Remember that the C message randomizer (which is included in the signature) has the same size (n bytes) as the hash output, and so it shrinks from 32 bytes to 24 bytes for the parameter sets that use either SHA-256/192 or SHAKE256/192.

4. Additional LMS Parameter Sets

The table below defines several many-time signature parameters called Leighton-Micali Signature (LMS) parameters, using the SHA-256/192, SHAKE256/256, and SHAKE256/192 hash functions:

Parameter Set Name	H	m	h	id
LMS_SHA256_M24_H5	SHA-256/192	24	5	0x0000000A

Parameter Set Name	H	m	h	id
LMS_SHA256_M24_H10	SHA-256/192	24	10	0x0000000B
LMS_SHA256_M24_H15	SHA-256/192	24	15	0x0000000C
LMS_SHA256_M24_H20	SHA-256/192	24	20	0x0000000D
LMS_SHA256_M24_H25	SHA-256/192	24	25	0x0000000E
LMS_SHAKE_M32_H5	SHAKE256/256	32	5	0x0000000F
LMS_SHAKE_M32_H10	SHAKE256/256	32	10	0x00000010
LMS_SHAKE_M32_H15	SHAKE256/256	32	15	0x00000011
LMS_SHAKE_M32_H20	SHAKE256/256	32	20	0x00000012
LMS_SHAKE_M32_H25	SHAKE256/256	32	25	0x00000013
LMS_SHAKE_M24_H5	SHAKE256/192	24	5	0x00000014
LMS_SHAKE_M24_H10	SHAKE256/192	24	10	0x00000015
LMS_SHAKE_M24_H15	SHAKE256/192	24	15	0x00000016
LMS_SHAKE_M24_H20	SHAKE256/192	24	20	0x00000017
LMS_SHAKE_M24_H25	SHAKE256/192	24	25	0x00000018

Table 2

Parameter Set Name: The human-readable name of the parameter set.

H: The second-preimage-resistant cryptographic hash function used within this parameter set.

m: The size in bytes of the hash function output.

h: The height of the Merkle tree.

id: The IANA-defined identifier used to denote this specific parameter set, which appears in both public keys and signatures.

These values are additions to the entries in Table 2 of [RFC8554].

The SHA256_M24, SHAKE_M32, and SHAKE_M24 in the parameter set names denote the SHA-256/192, SHAKE256/256, and SHAKE256/192 hash functions defined in Section 2.

5. Usage for These Additional Hash Functions within HSS

To use the additional hash functions within HSS, one would use the appropriate LM-OTS id from Table 1 and the appropriate LMS id from Table 2 and use that additional hash function when computing the hashes for key generation, signature generation, and signature verification.

Note that the size of the I Merkle tree identifier remains 16 bytes, independent of what hash function is used.

6. Parameter Set Selection

This document, along with [RFC8554], defines four hash functions for use within HSS/LMS: SHA-256, SHA-256/192, SHAKE256/256, and SHAKE256/192. The main reason one would select a hash with a 192-bit output (either SHA-256/192 or SHAKE256/192) would be to reduce the signature size; this comes at the cost of reducing the security margin. However, the security should be sufficient for most uses.

In contrast, there is no security or signature size difference between the SHA-256-based parameter sets (SHA-256 or SHA-256/192) versus the SHAKE-based parameter sets (SHAKE256/256 or SHAKE256/192). The reason for selecting between the two would be based on practical considerations, for example, if the implementation happens to have an existing SHA-256 (or SHAKE) implementation or if one of the two happens to give better hashing performance on the platform.

7. Comparisons of 192-Bit and 256-Bit Parameter Sets

Switching to a 192-bit hash affects the signature size, the computation time, and the security strength. It significantly reduces the signature size and somewhat reduces the computation time, at the cost of security strength. See Section 8 for a discussion of the security strength.

The impact on signature size and computation time is based on two effects:

1. Each hash that appears in the signature is shorter.
2. We need fewer Winternitz chains (because LM-OTS signs a shorter value).

For signature length, both effects are relevant. The first is relevant because the signature consists of a series of hashes and each hash is shorter. The second is relevant because when we need fewer Winternitz chains, we need fewer hashes in each LM-OTS signature.

For computation time (for both signature generation and verification), effect 1 is irrelevant (we still need to perform essentially the same hash computation), but effect 2 still applies. For example, with W=8, SHA-256 requires 34 Winternitz chains per LM-OTS signature, but SHA-256/192 requires only 26. Since the vast majority of time (for both signature generation and verification) is spent computing these Winternitz chains, this reduction in the number of chains gives us some performance improvement.

The table below gives the space used by both the 256-bit and 192-bit parameter sets for a range of commonly used Winternitz parameters and tree heights:

ParmSet	Winternitz	256-bit hash	192-bit hash
15	4	2672	1624
15	8	1616	1024
20	4	2832	1744
20	8	1776	1144
15/10	4	5236	3172
15/10	8	3124	1972
15/15	4	5396	3292
15/15	8	3284	2092
20/10	4	5396	3292
20/10	8	3284	2092
20/15	4	5556	3412
20/15	8	3444	2212

Table 3

ParmSet: The height of the Merkle tree(s), which is the parameter "h" from Table 2. Parameter sets listed as a single integer have L=1 and consist of a single Merkle tree of that height; parameter sets with L=2 are listed as x/y, with x being the height of the top-level Merkle tree and y being the bottom level.

Winternitz: The Winternitz parameter used, which is the parameter "w" from Table 1. For the tests that use multiple trees, this applies to all of them.

256-bit hash: The size in bytes of a signature, assuming that a 256-bit hash is used in the signature (either SHA-256 or SHAKE256/256).

192-bit hash: The size in bytes of a signature, assuming that a 192-bit hash is used in the signature (either SHA-256/192 or SHAKE256/192).

An examination of the signature sizes shows that the 192-bit parameters consistently give a 35-40% reduction in the size of the signature in comparison with the 256-bit parameters.

For SHA-256/192, there is a smaller (circa 20%) reduction in the amount of computation required for a signature operation with a 192-bit hash, because fewer Winternitz chains would need to be computed. The SHAKE256/192 signatures may have either a faster or slower computation, depending on the implementation speed of SHAKE versus SHA-256 hashes.

The SHAKE256/256-based parameter sets give no space advantage (or disadvantage) over the existing SHA-256-based parameter sets; any performance delta would depend solely on the implementation and whether they can generate SHAKE hashes faster than SHA-256 ones.

8. Security Considerations

The strength of a signature that uses the SHA-256/192, SHAKE256/256, and SHAKE256/192 hash functions is based on the difficulty in finding preimages or second preimages to those hash functions. As shown in [Katz16], if we assume that the hash function can be modeled as a random oracle, then the security of the system is at least 8N-1 bits (where N is the size of the hash output in bytes); this gives us a security level of 255 bits for SHAKE256/256 and 191 bits for SHA-256/192 and SHAKE256/192). That is, the strength of SHA-256/192 and SHAKE256/192 can be expected to be equivalent to the strength of AES-192, while the strength of SHAKE256/256 is equivalent to the strength of AES-256. If AES-192 and AES-256 are quantum-resistant, then we expect SHA-256/192, SHAKE256/192, and SHAKE256/256 to also be.

If we look at this in a different way, we see that the case of SHAKE256/256 is essentially the same as the existing SHA-256-based signatures; the difficultly of finding preimages and second preimages is essentially the same, and so they have (barring unexpected cryptographical advances) essentially the same level of security.

The case of SHA-256/192 and SHAKE256/192 requires closer analysis.

For a classical (non-quantum) computer, there is no known attack better than performing hashes of a large number of distinct preimages. Therefore, a successful attack has a high probability of requiring nearly 2^{192} hash computations (for either SHA-256/192 or SHAKE256/192). These can be taken as the expected work effort and would appear to be completely infeasible in practice.

In theory, an attacker with a quantum computer could use Grover's algorithm [Grover96] to reduce the expected complexity to circa 2^{96} hash computations (for N=24). On the other hand, implementing Grover's algorithm with this number of hash computations would require performing circa 2^{96} hash computations in succession, which will take more time than is likely to be acceptable to any attacker. To speed this up, the attacker would need to run a number of instances of Grover's algorithm in parallel. This would necessarily increase the total work effort required, and to an extent, that makes it likely infeasible. This is because if we limit the time taken by Grover's algorithm to 2^t steps (for t <= 96), then to attack a hash preimage problem of 192 bits, it requires a total of $2^{(192-t)}$ hash computations, rather than the $2^{(192/2)}$ hash computations it would require if we did not limit the time taken. In other words, the hash preimage can be found in 2^t steps by using $2^{(192-2t)}$ quantum computers (for t <= 96), with one of the quantum computers finding the preimage. For example, if the adversary is willing to wait for 2^{64} times the time taken by a hash computation (which is over 50 years if a quantum computer can compute a hash in 0.1 nanoseconds), this implies that a total of $2^{(192-64)} = 2^{128}$ hash computations will need to be performed, on 2^{64} (18 quintillion) separate quantum computers, each of which computes 2^{64} hash evaluations.

Hence, we expect that HSS/LMS based on these hash functions is secure against both classical and quantum computers, even though, in both cases, the expected work effort is less (for the N=24 case) than against either SHA-256 or SHAKE256/256.

SHA-256 is subject to a length extension attack. In this attack, if the attacker is given the hash value of an unknown message (and the message length), then the attacker can compute the hash of the message appended with certain strings (even though the attacker does not know the contents of the initial part of the modified message). This would appear to be irrelevant to HSS for two reasons:

- For the initial message hash, the hash is entirely on public data. Hence, this attack is irrelevant, because the attacker could compute the hash of the message with appended data anyways.
- The rest of the hashes within HSS are fixed length. Hence, there is no opportunity to perform length extension attacks.

In addition, to perform a length extension attack on SHA-256/192, the attacker has to guess the 64 omitted bits (because the attack requires all 256 bits of the hash value); hence, that is even less of a concern than it is for the standard SHA-256.

There is one corner case for which the security strength is reduced: if we need to assume that the signer will never deliberately generate a signature that is valid for two different messages. HSS uses randomized hashing when signing a message. That is, when a message is being presented to be signed, the signer generates a random value C and includes that in what is prepended to the message. Because the attacker cannot predict this value, it is infeasible for anyone other than the signer to find a generic collision. That is, practically speaking, a signature that is valid for two colliding messages is feasible only if the signer signed both messages. For this to happen, a signer (that is, the one with the private key and who picks the random C value) would have to break the collision resistance of the hash function to generate those two colliding messages. Note that this does not apply to someone who submits the messages for signing; only the signer could perform this. This would result in a signature that would be valid for two different selected messages. This is a nonstandard assumption for signature schemes and is usually not a concern, as we assume that the signer is trusted to generate signatures for any message. However, if the application needs to assume that it is infeasible for the signer to generate such a signature, then the security strength assumptions are reduced (128 bits for SHAKE256/256 and 96 bits for SHA-256/192 and SHAKE256/192).

Some cryptographers have raised the possibility of a multi-target attack (where the attacker has signatures from a large number of public keys and succeeds if they can generate a forgery against any one of those public keys). While no such method of attack has been proposed, the possibility cannot be excluded; if there are a large number of public keys, it might be prudent to consider the possibility of some security loss with N=24. If there are 2^K public keys, this security loss cannot be more than K bits of security.

8.1. Note on the Version of SHAKE

[FIPS202] defines both SHAKE128 and SHAKE256. This specification selects SHAKE256, even though it is less efficient for large messages. The reason is that SHAKE128 has a low upper bound on the difficulty of finding preimages (due to the invertibility of its internal permutation), which would limit the strength of HSS/LMS (whose strength is based on the difficulty of finding preimages). Hence, we specify the use of SHAKE256, which has a considerably stronger preimage resistance.

9. IANA Considerations

IANA has assigned the code points for the parameter sets in Section 3 in the "LM-OTS Signatures" registry and the parameter sets in Section 4 in the "Leighton-Micali Signatures (LMS)" registry. These assignments are included in [NIST_SP_800-208], but the IANA registrations only reference this document.

10. References

10.1. Normative References

[FIPS180] NIST, "Secure Hash Standard", NIST FIPS 180-4, DOI 10.6028/NIST.FIPS.180-4, August 2015, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf>.

[FIPS202] NIST, "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions", NIST FIPS 202, DOI 10.6028/NIST.FIPS.202, August 2015, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf>.

[RFC8554] 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>.

10.2. Informative References

[Grover96] Grover, L., "A fast quantum mechanical algorithm for database search", Proceedings of the twenty-eighth annual ACM symposium on Theory of Computing (STOC '96), pp. 212-219, DOI 10.1145/237814.237866, July 1996, <https://doi.org/10.1145/237814.237866>.

[Katz16] Katz, J., "Analysis of a Proposed Hash-Based Signature Standard", Security Standardisation Research (SSR 2016), Lecture Notes in Computer Science, vol. 10074, pp. 261-273, DOI 10.1007/978-3-319-49100-4_12, 2016, <https://doi.org/10.1007/978-3-319-49100-4_12>.

[NIST_SP_800-208] Cooper, D., Apon, D., Dang, Q., Davidson, M., Dworkin, M., and C. Miller, "Recommendation for Stateful Hash-Based Signature Schemes", National Institute of Standards and Technology, NIST SP 800-208, DOI 10.6028/NIST.SP.800-208, October 2020, <https://doi.org/10.6028/NIST.SP.800-208>.

Appendix A. Test Cases

This appendix provides four test cases that can be used to verify or debug an implementation. This data is formatted with the name of the elements on the left and the value of the elements on the right, in hexadecimal. The concatenation of all of the values within a public key or signature produces that public key or signature, and values that do not fit within a single line are listed across successive lines.

A.1. Test Case 1 - SHA-256/192

```
--------------------------------------------
(note: procedure in Appendix A of [RFC8554] is used)
SEED        000102030405060708090a0b0c0d0e0f
            1011121314151617
I           202122232425262728292a2b2c2d2e2f
--------------------------------------------
```

Figure 1: Private Key for SHA-256/192

```
--------------------------------------------
HSS public key
levels      00000001
--------------------------------------------
LMS type    0000000a                         # LMS_SHA256_M24_H5
LM-OTS type 00000008                         # LMOTS_SHA256_N24_W8
I           202122232425262728292a2b2c2d2e2f
K           2c571450aed99cfb4f4ac285da148827
            96618314508b12d2
--------------------------------------------
```

Figure 2: Public Key for SHA-256/192

```
--------------------------------------------
Message     54657374206d65737361676520666f72  |Test message for|
            20534841323536 2d3139320a          | SHA-256/192.|
--------------------------------------------
```

Figure 3: Message for SHA-256/192

```
--------------------------------------------
HSS signature
Nspk        00000000
sig[0]:
--------------------------------------------
LMS signature
q           00000005
--------------------------------------------
LM-OTS signature
LM-OTS type 00000008                          # LMOTS_SHA256_N24_W8
C           0b5040a18c1b5cabcbc85b047402ec62
            94a30dd8da8fc3da
y[0]        e13b9f0875f09361dc77fcc4481ea463
            c073716249719193
y[1]        614b835b4694c059f12d3aedd34f3db9
            3f3580fb88743b8b
y[2]        3d0648c0537b7a50e433d7ea9d6672ff
            fc5f42770feab4f9
y[3]        8eb3f3b23fd2061e4d0b38f832860ae7
            6673ad1a1a52a900
y[4]        5dcf1bfb56fe16ff723627612f9a48f7
            90f3c47a67f870b8
y[5]        1e919d99919c8db48168838cece0abfb
            683da48b9209868b
y[6]        e8ec10c63d8bf80d36498dfc205dc45d
            0dd870572d6d8f1d
y[7]        90177cf5137b8bbf7bcb67a46f86f26c
            fa5a44cbcaa4e18d
y[8]        a099a98b0b3f96d5ac8ac375d8da2a7c
            248004ba11d7ac77
y[9]        5b9218359cddab4cf8ccc6d54cb7e1b3
            5a36ddc9265c0870
y[10]       63d2fc6742a7177876476a324b03295b
            fed99f2eaf1f3897
y[11]       0583c1b2b616aad0f31cd7a4b1bb0a51
            e477e94a01bbb4d6
y[12]       f8866e2528a159df3d6ce244d2b6518d
            1f0212285a3c2d4a
y[13]       927054a1e1620b5b02aab0c8c10ed48a
            e518ea73cba81fcf
y[14]       ff88bff461dac51e7ab4ca75f47a6259
            d24820b9995792d1
y[15]       39f61ae2a8186ae4e3c9bfe0af2cc717
            f424f41aa67f03fa
y[16]       edb0665115f2067a46843a4cbbd297d5
            e83bc1aafc18d1d0
y[17]       3b3d894e8595a6526073f02ab0f08b99
            fd9eb208b59ff631
y[18]       7e5545e6f9ad5f9c183abd043d5acd6e
            b2dd4da3f02dbc31
y[19]       67b468720a4b8b92ddfe7960998bb7a0
            ecf2a26a37598299
y[20]       413f7b2aecd39a30cec527b4d9710c44
            73639022451f50d0
y[21]       1c0457125da0fa4429c07dad859c846c
            bbd93ab5b91b01bc
```

```
y[22]        770b089cfede6f651e86dd7c15989c8b
             5321dea9ca608c71
y[23]        fd862323072b827cee7a7e28e4e2b999
             647233c3456944bb
y[24]        7aef9187c96b3f5b79fb98bc76c3574d
             d06f0e95685e5b3a
y[25]        ef3a54c4155fe3ad817749629c30adbe
             897c4f4454c86c49
--------------------------------------------
LMS type     0000000a                         # LMS_SHA256_M24_H5
path[0]      e9ca10eaa811b22ae07fb195e3590a33
             4ea64209942fbae3
path[1]      38d19f152182c807d3c40b189d3fcbea
             942f44682439b191
path[2]      332d33ae0b761a2a8f984b56b2ac2fd4
             ab08223a69ed1f77
path[3]      19c7aa7e9eee96504b0e60c6bb5c942d
             695f0493eb25f80a
path[4]      5871cffd131d0e04ffe5065bc7875e82
             d34b40b69dd9f3c1
```

Figure 4: Signature for SHA-256/192

A.2. Test vector for SHAKE256/192

```
--------------------------------------------
(note: procedure in Appendix A of [RFC8554] is used)
SEED         303132333435363738393a3b3c3d3e3f
             4041424344454647
I            505152535455565758595a5b5c5d5e5f
--------------------------------------------
```

Figure 5: Private Key for SHAKE256/192

```
---------------------------------------------
HSS public key
levels       00000001
--------------------------------------------
LMS type     00000014                         # LMS_SHAKE_N24_H5
LM-OTS type  00000010                         # LMOTS_SHAKE_N24_W8
I            505152535455565758595a5b5c5d5e5f
K            db54a4509901051c01e26d9990e55034
             7986da87924ff0b1
--------------------------------------------
```

Figure 6: Public Key for SHAKE256/192

```
--------------------------------------------
Message      54657374206d65737361676520666f72  |Test message for|
             205348414b453235362d3139320a      | SHAKE256/192.|
--------------------------------------------
```

Figure 7: Message for SHAKE256/192

```
--------------------------------------------
HSS signature
Nspk        00000000
sig[0]:
--------------------------------------------
LMS signature
q           00000006
--------------------------------------------
LM-OTS signature
LM-OTS type 00000010                         # LMOTS_SHAKE_N24_W8
C           84219da9ce9fffb16edb94527c6d1056
            5587db28062deac4
y[0]        208e62fc4fbe9d85deb3c6bd2c01640a
            ccb387d8a6093d68
y[1]        511234a6a1a50108091c034cb1777e02
            b5df466149a66969
y[2]        a498e4200c0a0c1bf5d100cdb97d2dd4
            0efd3cada278acc5
y[3]        a570071a043956112c6deebd1eb3a7b5
            6f5f6791515a7b5f
y[4]        fddb0ec2d9094bfbc889ea15c3c7b9be
            a953efb75ed648f5
y[5]        35b9acab66a2e9631e426e4e99b733ca
            a6c55963929b77fe
y[6]        c54a7e703d8162e736875cb6a455d4a9
            015c7a6d8fd5fe75
y[7]        e402b47036dc3770f4a1dd0a559cb478
            c7fb1726005321be
y[8]        9d1ac2de94d731ee4ca79cff454c811f
            46d11980909f047b
y[9]        2005e84b6e15378446b1ca691efe491e
            a98acc9d3c0f785c
y[10]       aba5e2eb3c306811c240ba2280292382
            7d582639304a1e97
y[11]       83ba5bc9d69d999a7db8f749770c3c04
            a152856dc726d806
y[12]       7921465b61b3f847b13b2635a45379e5
            adc6ff58a99b00e6
y[13]       0ac767f7f30175f9f7a140257e218be3
            07954b1250c9b419
y[14]       02c4fa7c90d8a592945c66e86a76defc
            b84500b55598a199
y[15]       0faaa10077c74c94895731585c8f900d
            e1a1c675bd8b0c18
y[16]       0ebe2b5eb3ef8019ece3e1ea7223eb79
            06a2042b6262b4aa
y[17]       25c4b8a05f205c8befeef11ceff12825
            08d71bc2a8cfa0a9
y[18]       9f73f3e3a74bb4b3c0d8ca2abd0e1c2c
            17dafe18b4ee2298
y[19]       e87bcfb1305b3c069e6d385569a4067e
            d547486dd1a50d6f
y[20]       4a58aab96e2fa883a9a39e1bd45541ee
            e94efc32faa9a94b
y[21]       e66dc8538b2dab05aee5efa6b3b2efb3
            fd020fe789477a93
```

```
y[22]        afff9a3e636dbba864a5bffa3e28d13d
             49bb597d94865bde
y[23]        88c4627f206ab2b465084d6b780666e9
             52f8710efd748bd0
y[24]        f1ae8f1035087f5028f14affcc5fffe3
             32121ae4f87ac5f1
y[25]        eac9062608c7d87708f1723f38b23237
             a4edf4b49a5cd3d7
--------------------------------------------
LMS type     00000014                          # LMS_SHAKE_N24_H5
path[0]      dd4bdc8f928fb526f6fb7cdb944a7eba
             a7fb05d995b5721a
path[1]      27096a5007d82f79d063acd434a04e97
             f61552f7f81a9317
path[2]      b4ec7c87a5ed10c881928fc6ebce6dfc
             e9daae9cc9dba690
path[3]      7ca9a9dd5f9f573704d5e6cf22a43b04
             e64c1ffc7e1c442e
path[4]      cb495ba265f465c56291a902e62a461f
             6dfda232457fad14
```

Figure 8: Signature for SHAKE256/192

A.3. Test vector for SHA-256/256

```
--------------------------------------------
(note: procedure in Appendix A of [RFC8554] is used)
SEED         606162636465666768696a6b6c6d6e6f
             707172737475767778797a7b7c7d7e7f
I            808182838485868788898a8b8c8d8e8f
--------------------------------------------
```

Figure 9: Private Key for SHAKE256/256

```
--------------------------------------------
HSS public key
levels       00000001
--------------------------------------------
LMS type     0000000f                          # LMS_SHAKE_N32_H5
LM-OTS type  0000000c                          # LMOTS_SHAKE_N32_W8
I            808182838485868788898a8b8c8d8e8f
K            9bb7faee411cae806c16a466c3191a8b
             65d0ac31932bbf0c2d07c7a4a36379fe
--------------------------------------------
```

Figure 10: Public Key for SHAKE256/256

```
--------------------------------------------
Message     54657374206d657373616765206f7220  |Test message for|
            5348414b453235362d3235360a        |SHAKE256/256.|
--------------------------------------------
```

Figure 11: Message for SHAKE256/256

```
--------------------------------------------
HSS signature
Nspk        00000000
sig[0]:
--------------------------------------------
LMS signature
q           00000007
--------------------------------------------
LM-OTS signature
LM-OTS type 0000000c                         # LMOTS_SHAKE_N32_W8
C           b82709f0f00e83759190996233d1ee4f
            4ec50534473c02ffa145e8ca2874e32b
y[0]        16b228118c62b96c9c77678b33183730
            debaade8fe607f05c6697bc971519a34
y[1]        1d69c00129680b67e75b3bd7d8aa5c8b
            71f02669d177a2a0eea896dcd1660f16
y[2]        864b302ff321f9c4b8354408d0676050
            4f768ebd4e545a9b0ac058c575078e6c
y[3]        1403160fb45450d61a9c8c81f6bd69bd
            fa26a16e12a265baf79e9e233eb71af6
y[4]        34ecc66dc88e10c6e0142942d4843f70
            a0242727bc5a2aabf7b0ec12a99090d8
y[5]        caeef21303f8ac58b9f200371dc9e41a
            b956e1a3efed9d4bbb38975b46c28d5f
y[6]        5b3ed19d847bd0a737177263cbc1a226
            2d40e80815ee149b6cce2714384c9b7f
y[7]        ceb3bbcbd25228dda8306536376f8793
            ecadd6020265dab9075f64c773ef97d0
y[8]        7352919995b74404cc69a6f3b469445c
            9286a6b2c9f6dc839be76618f053de76
y[9]        3da3571ef70f805c9cc54b8e501a98b9
            8c70785eeb61737eced78b0e380ded4f
y[10]       769a9d422786def59700eef3278017ba
            bbe5f9063b468ae0dd61d94f9f99d5cc
y[11]       36fbec4178d2bda3ad31e1644a2bcce2
            08d72d50a7637851aa908b94dc437612
y[12]       0d5beab0fb805e1945c41834dd6085e6
            db1a3aa78fcb59f62bde68236a10618c
y[13]       ff123abe64dae8dabb2e84ca705309c2
            ab986d4f8326ba0642272cb3904eb96f
y[14]       6f5e3bb8813997881b6a33cac0714e4b
            5e7a882ad87e141931f97d612b84e903
y[15]       e773139ae377f5ba19ac86198d485fca
            97742568f6ff758120a89bf19059b8a6
y[16]       bfe2d86b12778164436ab2659ba86676
            7fcc435584125fb7924201ee67b535da
y[17]       f72c5cb31f5a0b1d926324c26e67d4c3
            836e301aa09bae8fb3f91f1622b1818c
y[18]       cf440f52ca9b5b9b99aba8a6754aae2b
            967c4954fa85298ad9b1e74f27a46127
y[19]       c36131c8991f0cc2ba57a15d35c91cf8
            bc48e8e20d625af4e85d8f9402ec44af
y[20]       bd4792b924b839332a64788a7701a300
            94b9ec4b9f4b648f168bf457fbb3c959
y[21]       4fa87920b645e42aa2fecc9e21e000ca
            7d3ff914e15c40a8bc533129a7fd3952
```

```
y[22]       9376430f355aaf96a0a13d13f2419141
            b3cc25843e8c90d0e551a355dd90ad77
y[23]       0ea7255214ce11238605de2f000d2001
            04d0c3a3e35ae64ea10a3eff37ac7e95
y[24]       49217cdf52f307172e2f6c7a2a4543e1
            4314036525b1ad53eeaddf0e24b1f369
y[25]       14ed22483f2889f61e62b6fb78f5645b
            dbb02c9e5bf97db7a0004e87c2a55399
y[26]       b61958786c97bd52fa199c27f6bb4d68
            c4907933562755bfec5d4fb52f06c289
y[27]       d6e852cf6bc773ffd4c07ee2d6cc55f5
            7edcfbc8e8692a49ad47a121fe3c1b16
y[28]       cab1cc285faf6793ffad7a8c341a49c5
            d2dce7069e464cb90a00b2903648b23c
y[29]       81a68e21d748a7e7b1df8a593f3894b2
            477e8316947ca725d141135202a9442e
y[30]       1db33bbd390d2c04401c39b253b78ce2
            97b0e14755e46ec08a146d279c67af70
y[31]       de256890804d83d6ec5ca3286f1fca9c
            72abf6ef868e7f6eb0fddda1b040ecec
y[32]       9bbc69e2fd8618e9db3bdb0af13dda06
            c6617e95afa522d6a2552de15324d991
y[33]       19f55e9af11ae3d5614b564c642dbfec
            6c644198ce80d2433ac8ee738f9d825e
--------------------------------------------
LMS type    0000000f                         # LMS_SHAKE_N32_H5
path[0]     71d585a35c3a908379f4072d070311db
            5d65b242b714bc5a756ba5e228abfa0d
path[1]     1329978a05d5e815cf4d74c1e547ec4a
            a3ca956ae927df8b29fb9fab3917a7a4
path[2]     ae61ba57e5342e9db12caf6f6dbc5253
            de5268d4b0c4ce4ebe6852f012b162fc
path[3]     1c12b9ffc3bcb1d3ac8589777655e22c
            d9b99ff1e4346fd0efeaa1da044692e7
path[4]     ad6bfc337db69849e54411df8920c228
            a2b7762c11e4b1c49efb74486d3931ea
```

Figure 12: Signature for SHAKE256/256

A.4. Test vector for SHA-256/192, W=4

```
--------------------------------------------
(note: procedure in Appendix A of [RFC8554] is used)
SEED        202122232425262728292a2b2c2d2e2f
            3031323334353637
I           404142434445464748494a4b4c4d4e4f
--------------------------------------------
```

Figure 13: Private Key for SHA256/192 with W=4

```
--------------------------------------------
HSS public key
levels      00000001
--------------------------------------------
LMS type    0000000d                         # LMS_SHA256_M24_H20
LM-OTS type 00000007                         # LMOTS_SHA256_N24_W4
I           404142434445464748494a4b4c4d4e4f
K           9c08a50d170406869892802ee4142fcd
            eac990f110c2460c
--------------------------------------------
```

Figure 14: Public Key for SHA256/192 with W=4

```
--------------------------------------------
Message     54657374206d65737361676520666f72  |Test message for|
            205348413235362f31393220773d34    | SHA256/192 w=4|
--------------------------------------------
```

Figure 15: Message for SHA256/192 with W=4

```
--------------------------------------------
HSS signature
Nspk        00000000
sig[0]:
--------------------------------------------
LMS signature
q           00000064
--------------------------------------------
LM-OTS signature
LM-OTS type 00000007                          # LMOTS_SHA256_N24_W4
C           853fa6e1a65fef076acd2485505b93be
            9aeb2641e3d3805c
y[0]        1887f26f4bcdb6ac0337b76fa5d66038
            34287e010b20516f
y[1]        7c336df2134c0a981f1ec2bb7baee516
            e91e67d3bd16c8d9
y[2]        45a7f2be4fd84a604ae3743efc609ee0
            e69572e9c6d4a682
y[3]        50e877b75d3cae63e9d5c15a32bb3cd1
            7045f6b3e195284f
y[4]        dd1ee3cfbe18f1cbd06ef3e7af34b184
            4d42dac453115a45
y[5]        07ed525cec120d054b403c61a7e5034f
            ac4be6ef5412d194
y[6]        d4b6bbc0ae6cd3fe9993d583ee06f403
            0bc832efec24d1f7
y[7]        13f5088731b91a98491fa3adf1b322bc
            e26df24c8415e3a4
y[8]        6bdfe07a6fd48e6d951515758cd64349
            91098bf6949249fc
y[9]        a338ec235871dd564998d07d9b1b1b8d
            644e657fee8039da
y[10]       8fe195d129faddb12d543b86b0ab8cf6
            f26c121783f3b828
y[11]       d03f793b42909272f688e4ef6d46e82b
            dd1a02b1ff86c3b7
y[12]       9920b2e6f19faf75c623242f1f2c549f
            84fb2f4c3ffead31
y[13]       20d97baea507467bb2da79f132bbe15b
            596fdfcb70983107
y[14]       ebca2597de9d55bd83bcae5c28a85259
            dadb354859986e60
y[15]       c8afa0b10bd08a8f9ed9b1ede3377075
            fe0ae36349f7d2ed
y[16]       7bfc9ece0d4cd6972059329419feaf3b
            9a1045b6cfa4ae89
y[17]       b1cea8950aea4af870d1a3a3909ebc5a
            3013d6deb927abc0
y[18]       f95093e83cb36a9c1d6f13add19268ac
            7a0371f8335b0952
y[19]       a57fdb0141d55d937dd6ebb08fee8a5c
            f426ac97d54ee7aa
y[20]       17e6c57be5e62a52a6b1b986730d3a3a
            ad8a7d327ddf883e
y[21]       6bc7b636eb2a5c4f2a635ae5bada5418
            d43dfedb69c0a020
```

```
y[22]        9334fac89d420d6ad5a2e1df95d26a1b
             feb99a5e8455061b
y[23]        fdf2d6e8394caf8a4be699b8afa38e52
             4d4053330af478f8
y[24]        5bf33d3ca3a35bc96987282bd513a8f6
             a52db9ba36aa9088
y[25]        2b3bf573fa275449d8d49eb30bed2bb1
             7a0ecc7d8a20807f
y[26]        2ea3dd37acd46c713cc2ac9d01a20a30
             d6832eef86a1e26d
y[27]        1cad7761bf4130a6565572766026509d
             eeddaf46b605452b
y[28]        218a4e137a7ce063b546a35c52510f0e
             a2cac879192ec443
y[29]        e43b37c5ffa23da7a7fc254324a3de70
             5c771794f10ea356
y[30]        e5a747e5146fd804a47719803c185b38
             0e34b8dcc8269c2b
y[31]        073d86b2307cf90c6c3ef9271f2d53df
             2579f0c4cfb632db
y[32]        37a9025965f70b4616673228e98644be
             6576417b7a97f104
y[33]        350259e7f697408cdf8cf81a3e774162
             6ccdb87ad8531264
y[34]        cb5ceb7c8c097cec505091a3ee3a826c
             54f78169abc2e7d0
y[35]        a318dac10250ba940e51e79a3f572fb3
             2bf442be6fd81267
y[36]        946e6387f9a8c705d945c653f2684655
             e3fa6b9ee311d8a0
y[37]        91bef9898292fa272fb8761f066c23d8
             7aa10d67871cc541
y[38]        9c843b796855c51ad1272e9264acd203
             5a82b12c2ddbc85a
y[39]        dfcd7c22366a36495349391dbf000106
             4b8f6b28365445d7
y[40]        33e48f1b058a6cb3e71bbb8df3e90406
             299894f4ca682943
y[41]        ceeba410b33b07716ffc18d6eab75f2d
             6372f1133605fa3c
y[42]        3ed66f2d8f7c5abe59e87d4500965e34
             7523d73cb356c144
y[43]        827aaa22b1c72a15293c7400e02aaefc
             f36f68a8246900e6
y[44]        e6228e7ad19d1450c23434f1e45043dc
             2b6db57f20d8f5b3
y[45]        44d4162aa651333287cd8bf8fac41c78
             d61fe2929209bfe2
y[46]        dc5a2f80205c043b22e540a29f0ea0a5
             ff529e55bf1dfe42
y[47]        96fc4bb4ac2e875322ab115db479fe97
             9d64f78409af4ec3
y[48]        ad3b758fff83af1b9c48e90ca39366f4
             26c2fb921df55c72
y[49]        786a9217723945a1ac1a66af7def4f8b
             367001732cce0e5b
y[50]        ac91ac9d603807f8bab105b46d315d4c
             b88feb1c8686884b
```

```
--------------------------------------------
LMS type    0000000d                          # LMS_SHA256_M24_H20
path[0]    13d1a8ef00c5811c15c4d774fdcf7515
           5315aff53ebdff8f
path[1]    b6a54f12c165963dd5690cc9842b0e21
           90afc5443497584c
path[2]    832155599d00aced84bb3b59170396f7
           db4fa84aa8577f76
path[3]    cf9367d6e99d3d5be3555d7156b004f2
           002f505681b1ad22
path[4]    9b9b46a666672aa8ee662c3a0456a9ad
           da7a44fbaca46789
path[5]    577dcd36dc5cdff34b864d0a32492a0a
           cbcaa6c011748f20
path[6]    5b91ab2ab84f2333fb3e3b9acaecdac3
           8b58aa5f32e718e2
path[7]    25631ed6674cccb8c119acbd4992ab31
           30a6e912deec5983
path[8]    5ab52fbc549430f8b403e4a2a51cc7f4
           6fc143d365763aa1
path[9]    708fd25bcd657a790e54718d97090624
           2a3b8a97dff18e91
path[10]   a44c4ba818a8dd2d242251265b023b82
           6077eb740f6682e6
path[11]   c4ada2b85a67988d406132c2ad899099
           e44cfe610c3a5af7
path[12]   0b406224411a59597e5dda0f31cd16c9
           14b67e96141661f0
path[13]   074f43eb02273481bc324ded26c64f23
           88559d8c8bd0ef8b
path[14]   34ca4afebfac2a689b4246c264241488
           dcf922350dc44f7b
path[15]   c09d57dc1126291b2318810e0f44801c
           071e572fd032c780
path[16]   f44c9503a4c03c37417dc96422ba0849
           c37956f9fd5d33ea
path[17]   4fcab84276effec652ca77d7d47ac93c
           633d99e0a236f03d
path[18]   5587d1990ffaef737fced1f5cdd8f373
           844e9f316aad41a0
path[19]   b12302639f83a2d74c9fe30d305a942b
           c0c30352a5e44dfb
```

Figure 16: Signature for SHA256/192 with W=4

Acknowledgements

We would like to thank Carsten Bormann, Russ Housley, Andrey Jivsov, Mallory Knodel, Virendra Kumar, Thomas Pornin, and Stanislav Smyshlyaev for their insightful and helpful reviews.

Authors' Addresses

Scott Fluhrer
Cisco Systems
170 West Tasman Drive
San Jose, CA
United States of America
Email: sfluhrer@cisco.com

Quynh Dang
NIST
100 Bureau Drive
Gaithersburg, MD
United States of America
Email: quynh.dang@nist.gov