Crypto Forum Research Group S. Fluhrer
Internet-Draft Cisco Systems
Intended status: Informational Q. Dang
Expires: September 20, 2020 NIST
March 19, 2020

Additional Parameter sets for LMS Hash-Based Signatures
draft-fluhrer-lms-more-parm-sets-01

Abstract

This note extends LMS (RFC 8554) by defining parameter sets by including additional hash functions. Hese include hash functions that result in signatures with significantly smaller than the signatures using the current parameter sets, and should have sufficient security.

This document is a product of the Crypto Forum Research Group (CFRG) in the IRTF.

Status of This Memo

Copyright Notice

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 1k - 4kbytes). What this draft explores are a set of parameter sets to the LMS (RFC8554) stateful hash based signature method that reduce the size of the signature significantly.

1.1. Disclaimer

This document is not intended as legal advice. Readers are advised to consult with their own legal advisers if they would like a legal interpretation of their rights.

The IETF policies and processes regarding intellectual property and patents are outlined in [RFC3979] and [RFC4879] and at https://datatracker.ietf.org/ipr/about.

2. Conventions Used In This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Additional Hash Function Definitions

3.1. 192 bit Hash Function based on SHA256

This document defines a SHA-2 based hash function with a 192 bit output. As such, we define SHA256/192 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. It is the same procedure used to define SHA-224, except that we use the SHA-256 IV (rather than using one dedicated to SHA256/192), and you truncate 64 bits, rather than 32.

The following test vector may illustrate this:

```
SHA256("abc")     = ba7816bf 8f01cfea 414140de 5dae2223
                    b00361a3 96177a9c b410ff61 f20015ad
SHA256/192("abc") = ba7816bf 8f01cfea 414140de 5dae2223
                    b00361a3 96177a9c
```

We use the same IV 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 you get partial knowledge of the SHA-256 hash of a message by examining the SHA256/192 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 which varies per message; in addition, all hashes include the I identifier, which varies depending on the public key. Therefore, signing the same message by SHA256 and by SHA256/192 will not result in the same value being hashed, and so the latter hash value is not a prefix of the former one.

3.2. 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 as a hash where you submit the preimage to the SHAKE256 XOF, with the output being 256 bits, see FIPS 202 [FIPS202] for more detail.

3.3. 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 as a hash where you submit the preimage to the SHAKE-256 XOF, with the output being 192 bits, see FIPS 202 [FIPS202] for more detail.

4. Additional LM-OTS Parameter Sets

Here is a table with the LM-OTS parameters defined that use the above hashes:

Parameter Set Name	H	n	w	p	ls	id
LMOTS_SHA256_N24_W1	SHA256/192	24	1	200	8	TBD1
LMOTS_SHA256_N24_W2	SHA256/192	24	2	101	6	TBD2
LMOTS_SHA256_N24_W4	SHA256/192	24	4	51	4	TBD3
LMOTS_SHA256_N24_W8	SHA256/192	24	8	26	0	TBD4
LMOTS_SHAKE_N32_W1	SHAKE256-256	32	1	265	7	TBD5
LMOTS_SHAKE_N32_W2	SHAKE256-256	32	2	133	6	TBD6
LMOTS_SHAKE_N32_W4	SHAKE256-256	32	4	67	4	TBD7
LMOTS_SHAKE_N32_W8	SHAKE256-256	32	8	34	0	TBD8
LMOTS_SHAKE_N24_W1	SHAKE256-192	24	1	200	8	TBD9
LMOTS_SHAKE_N24_W2	SHAKE256-192	24	2	101	6	TBD10
LMOTS_SHAKE_N24_W4	SHAKE256-192	24	4	51	4	TBD11
LMOTS_SHAKE_N24_W8	SHAKE256-192	24	8	26	0	TBD12

Table 1

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

The SHA256_N24, SHAKE_N32, SHAKE_N24 in the parameter set name denote the SHA256/192, SHAKE256-256 and SHAKE256-192 hash functions defined in Section 3.

Remember that the C message randomizer (which is included in the signature) is the size of the hash n, and so it shrinks from 32 bytes to 24 bytes for those the parameter sets that use either SHA256/192 or SHAKE256-192.

5. Additional LM Parameter Sets

Here is a table with the LM parameters defined that use SHA259/192, SHAKE256-256 and SHAKE256-192 hash functions:

Parameter Set Name	H	m	h	id
LMS_SHA256_M24_H5	SHA256/192	24	5	TBD13
LMS_SHA256_M24_H10	SHA256/192	24	10	TBD14
LMS_SHA256_M24_H15	SHA256/192	24	15	TBD15
LMS_SHA256_M24_H20	SHA256/192	24	20	TBD16
LMS_SHA256_M24_H25	SHA256/192	24	25	TBD17
LMS_SHAKE_M32_H5	SHAKE256-256	32	5	TBD18
LMS_SHAKE_M32_H10	SHAKE256-256	32	10	TBD19
LMS_SHAKE_M32_H15	SHAKE256-256	32	15	TBD20
LMS_SHAKE_M32_H20	SHAKE256-256	32	20	TBD21
LMS_SHAKE_M32_H25	SHAKE256-256	32	25	TBD22
LMS_SHAKE_M24_H5	SHAKE256-192	24	5	TBD23
LMS_SHAKE_M24_H10	SHAKE256-192	24	10	TBD24
LMS_SHAKE_M24_H15	SHAKE256-192	24	15	TBD25
LMS_SHAKE_M24_H20	SHAKE256-192	24	20	TBD26
LMS_SHAKE_M24_H25	SHAKE256-192	24	25	TBD27

Table 2

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

The SHA256_M24, SHAKE_M32, SHAKE_M24 in the parameter set name denote the SHA256/192, SHAKE256-256 and SHAKE256-192 hash functions defined in Section 3.

6. 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.

The major reason for considering these truncated parameter sets is that they cause the signatures to shrink considerably.

Here is a table that gives the space used by both the 256 bit parameter sets and the 192 bit parameter sets, for a range of plausible 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: this is the height of the Merkle tree(s); parameter sets listed as a single integer have L=1, and consist 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: this is the Winternitz parameter used (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 SHA256 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 SHA256/192 or SHAKE256/192).

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

In addition, for SHA256-192, there is a smaller (circa 20%) reduction in the amount of computation required for a signature operation with a 192 bit hash. 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 SHA256-based parameter sets; any performance delta would depend solely on the implementation and whether they can generate SHAKE hashes faster than SHA-256 ones.

7. IANA Considerations

[TO BE REMOVED: The entries from Section 4, namely LMOTS_SHA256_N24_W1 through LMOTS_SHAKE_N24_W8 , should be inserted into https://www.iana.org/assignments/leighton-micali-signatures/leighton-micali-signatures.xhtml#lm-ots-signatures]

[TO BE REMOVED: The entries from Section 5, namely LMS_SHA256_M24_H5 through LMS_SHAKE_M24_H25 should be inserted into https://www.iana.org/assignments/leighton-micali-signatures/leighton-micali-signatures.xhtml#leighton-micali-signatures-1]

Until IANA assigns the codepoints, we will (for testing purposes only) use the following private use code points to do any necessary interoperability testing. Such an implementation must change to the IANA-assigned code points when they become available.

Parameter Set Name	Temporary Codepoint

LMOTS_SHA256_N24_W1	0xE0000001
LMOTS_SHA256_N24_W2	0xE0000002
LMOTS_SHA256_N24_W4	0xE0000003
LMOTS_SHA256_N24_W8	0xE0000004
LMOTS_SHAKE_N32_W1	0xE0000005
LMOTS_SHAKE_N32_W2	0xE0000006
LMOTS_SHAKE_N32_W4	0xE0000007
LMOTS_SHAKE_N32_W8	0xE0000008
LMOTS_SHAKE_N24_W1	0xE0000009
LMOTS_SHAKE_N24_W2	0xE000000A
LMOTS_SHAKE_N24_W4	0xE000000B
LMOTS_SHAKE_N24_W8	0xE000000C
LMS_SHA256_M24_H5	0xE0000001
LMS_SHA256_M24_H10	0xE0000002
LMS_SHA256_M24_H15	0xE0000003
LMS_SHA256_M24_H20	0xE0000004
LMS_SHA256_M24_H25	0xE0000005
LMS_SHAKE_M32_H5	0xE0000006
LMS_SHAKE_M32_H10	0xE0000007
LMS_SHAKE_M32_H15	0xE0000008
LMS_SHAKE_M32_H20	0xE0000009
LMS_SHAKE_M32_H25	0xE000000A
LMS_SHAKE_M24_H5	0xE000000B
LMS_SHAKE_M24_H10	0xE000000C

LMS_SHAKE_M24_H15	0xE000000D
LMS_SHAKE_M24_H20	0xE000000E
LMS_SHAKE_M24_H25	0xE000000F

Table 4

8. Security Considerations

The strength of a signature that uses the SHA256/192, SHAKE256-256 and SHAKE256-192 hash functions is based on the difficultly in finding preimages or second preimages to those hash functions.

The case of SHAKE256-256 is essentially the same as the existing SHA-256 based signatures; the difficultly of finding preimages is essentially the same, and so they have (barring unexpected cryptographical advances) essentially the same level of security.

The case of SHA256/192 and SHAKE256-192 requires closer analysis.

For a classical (nonquantum) computer, they have no known attack better than performing hashes of a large number of distinct preimages; as a successful attack has a high probability of requiring nearly 2**192 hash computations (for either SHA256/192 or SHAKE256-192). These can be taken as the expected work effort, and would appear to be completely infeasible in practice.

For a Quantum Computer, they could in theory use a Grover's algorithm to reduce the expected complexity required to circa 2**96 hash computations (for N=24). On the other hand, to implement 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 to be infeasible.

Hence, we expect that 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 SHA256 or SHAKE256-256.

8.1. Note on the version of SHAKE

FIPS 202 defines both SHAKE-128 and SHAKE-256. This specification selects SHAKE-256, even though it is, for large messages, less efficient. The reason is that SHAKE-128 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 LMS (whose strength is based on the difficulty of finding preimages). Hence, we specify the use of SHAKE-256, which has a considerably stronger preimage resistance.

9. References

9.1. Normative References

[FIPS180] National Institute of Standards and Technology, "Secure Hash Standard (SHS)", FIPS 180-4, March 2012.

[FIPS202] National Institute of Standards and Technology, "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions", FIPS 202, August 2015.

[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>.

[RFC3979] Bradner, S., Ed., "Intellectual Property Rights in IETF Technology", RFC 3979, DOI 10.17487/RFC3979, March 2005, <https://www.rfc-editor.org/info/rfc3979>.

[RFC4879] Narten, T., "Clarification of the Third Party Disclosure Procedure in RFC 3979", RFC 4879, DOI 10.17487/RFC4879, April 2007, <https://www.rfc-editor.org/info/rfc4879>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <https://www.rfc-editor.org/info/rfc5226>.

[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>.

9.2. Informative References

[Grover96]
Grover, L., "A fast quantum mechanical algorithm for database search", 28th ACM Symposium on the Theory of Computing p. 212, 1996.

Appendix A. Test Cases

In the future, this section will include an example test vector that uses the new hash functions

Authors' Addresses

Scott Fluhrer
Cisco Systems
170 West Tasman Drive
San Jose, CA
USA

Email: sfluhrer@cisco.com

Quynh Dang
NIST
100 Bureau Drive
Gaithersburg, MD
USA

Email: quynh.dang@nist.gov