Workgroup: HIP
Internet-Draft:
draft-moskowitz-hip-new-crypto-05
Updates: 7401 (if approved)
Published: 26 July 2020
Intended Status: Standards Track
Expires: 27 January 2021
Authors: R. Moskowitz HTT Consulting
S. Card AX Enterprize
A. Wiethuechter AX Enterprize

# New Cryptographic Algorithms for HIP

## Abstract

This document provides new cryptographic algorithms to be used with HIP. The Edwards Elliptic Curve and the Keccak sponge functions are the main focus. The HIP parameters and processing instructions impacted by these algorithms are defined.

## 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 27 January 2021.

**Table of Contents**



## 1. Introduction

This document adds new cryptographic algorithms for HIPv2 [RFC7401]. This includes:

- New elliptic curves for ECDH.
- The Edwards Elliptic Curve Digital Signature Algorithm (EdDSA) used in Host Identities (HI) and for Base Exchange (BEX) signatures.
- Hashes used in Host Identity Tag (HIT) generation, and wherever else hashes are needed.
- Keyed hashes used for KEYMAT generation and packet MACing operations.
- AEAD and stream ciphers to use in HIP and HIP enabled secure communication protocols.

The hashes and encryption are all built on the [Keccak] sponge function.

These additions reflect selection of advances in the field of cryptography that would best benefit HIP, particularly in constrained devices and communications.

## 2. Terms and Definitions

### 2.1. Requirements 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.

### 2.2. Definitions

**Keccak (KECCAK Message Authentication Code):**
The family of all sponge functions with a KECCAK-f permutation as the underlying function and multi-rate padding as the padding rule.

**KMAC (KECCAK Message Authentication Code):**
A PRF and keyed hash function based on KECCAK.

**cSHAKE (The customizable SHAKE function):**
Extends the SHAKE scheme to allow users to customize their use of the function.

**SHAKE (Secure Hash Algorithm KECCAK):**
A secure hash that allows for an arbitrary output length.

**PRF (Pseudorandom Function):**
A function that can be used to generate output from a random seed such that the output is computationally indistinguishable from truly random output.

**capacity:**
In the sponge construction, the width of the underlying function minus the rate.

**rate:**
In the sponge construction, the number of input bits processed per invocation of the underlying function.

**XOF (eXtendable-Output Function):**
A function on bit strings (also called messages) in which the output can be extended to any desired length.

## 3. HIP Parameter values for new Crytpo

HIP parameters carry information that is necessary for establishing and maintaining a HIP association. For example, the device's public keys as well as the signaling for negotiating ciphers and payload handling are encapsulated in HIP parameters. Additional information, meaningful for end hosts or middleboxes, may also be included in HIP parameters. The specification of the HIP parameters and their mapping to HIP packets and packet types is flexible to allow HIP extensions to define new parameters and new protocol behavior.

### 3.1. Elliptic Curves for Diffie-Hellman

Elliptic curves Curve25519 and Curve448 [RFC7748] are specified here for use in the HIP Diffie-Hellman exchange.

Curve25519 and Curve448 are already defined in Section 5.2.1 of [I-D.ietf-hip-dex], using the HIP-DEX CKDF. Here they are defined for using the new KMAC [NIST.SP.800-185] derived KDF in Section 5.

#### 3.1.1. DIFFIE_HELLMAN

The DIFFIE_HELLMAN parameter may be included in selected HIP packets based on the DH Group ID selected. The DIFFIE_HELLMAN parameter is defined in Section 5.2.7 of [RFC7401].

The following Elliptic Curves are defined here:

| Group | KDF | Value |
|---|---|---|
| Curve25519 [RFC7748] | KKDF | 13 |
| Curve448 [RFC7748] | KKDF | 14 |

A new KDF for KEYMAT, Section 6.5 of [RFC7401] and Section 6.3 of [I-D.ietf-hip-dex] using Keccak is defined in Section 5.

### 3.2. Edward Digital Signature Algorithm

Edwards-Curve Digital Signature Algorithm (EdDSA) [RFC8032] are specified here for use as Host Identities (HIs).

#### 3.2.1. HOST_ID

The HOST_ID parameter specifies the public key algorithm, and for elliptic curves, a name. The HOST_ID parameter is defined in Section 5.2.19 of [RFC7401].

| Algorithm profiles | Values | |
|---|---|---|
| EdDSA | 13 [RFC8032] | (RECOMMENDED) |

For hosts that implement EdDSA as the algorithm, the following ECC curves are available:

| Algorithm | Curve | Values |
|---|---|---|
| EdDSA | RESERVED | 0 |
| EdDSA | EdDSA25519 | 1 [RFC8032] |
| EdDSA | EdDSA25519ph | 2 [RFC8032] |
| EdDSA | EdDSA448 | 3 [RFC8032] |
| EdDSA | EdDSA448ph | 4 [RFC8032] |

### 3.2.2. HIT_SUITE_LIST

The HIT_SUITE_LIST parameter contains a list of the supported HIT suite IDs of the Responder. Based on the HIT_SUITE_LIST, the Initiator can determine which source HIT Suite IDs are supported by the Responder. The HIT_SUITE_LIST parameter is defined in Section 5.2.10 of [RFC7401].

The following HIT Suite ID is defined, and the relationship between the four-bit ID value used in the OGA ID field and the eight-bit encoding within the HIT_SUITE_LIST ID field is clarified:

| HIT Suite | Four-bit ID | Eight-bit encoding | |
|---|---|---|---|
| RESERVED | 0 | 0x00 | |
| EdDSA/cSHAKE128 | 5 | 0x50 | (RECOMMENDED) |

The following table provides more detail on the above HIT Suite combinations. The input for each generation algorithm is the encoding of the HI as defined in Section 4. The output is 96 bits long and is directly used in the ORCHID.

| Index | Hash function | HMAC | Signature algorithm family | Description |
|---|---|---|---|---|
| 5 | cSHAKE128 | KMAC128 | EdDSA | EdDSA HI hashed with cSHAKE128, output is 96 bits |

Table 1: HIT Suites

### 3.3. Hashing with the Keccak Function

The [Keccak] sponge function is the basis for the new SHA-3, standard [NIST.FIPS.202], and the customized XOF functions in [NIST.SP.800-185]. These are used here as an alternative to all the hashing functions in HIP.

Hardware implementation of Keccak in VHDL is available from [Keccak].

#### 3.3.1. The Keccak Permutation

Keccak is described as a sponge function. The analogy to a sponge is that an arbitrary number of input bits are "absorbed" into the state of the function, after which an arbitrary number of output bits are "squeezed" out of its state.

> The Keccak function is defined to have a width of b bits. Where b is the capacity (c) + rate (r).
>
> The rate is the number of bits "fed" into the sponge at a time.
>
> The capacity is twice the desired hash "strength" and part of the sponge width.

b is one of the set {25, 50, 100, 200, 400, 800, 1600}. In FIPS 202, b=1600. Thus a hash strength of 128 bits can be delivered with c=256 and r=1344, or 168 byte segment input to the sponge.

Keccak can also provide a hash strength of 128 bit with b=800 (r=544 or 68 bytes) and b=400 (r=144 or 18 bytes). 256 bit strength can only be provided with b=1600 or 800.

FIPS 202 does not specify use of these smaller values for b which may be preferred in memory constrained devices, processing relatively short input strings. Future work will determine if the smaller values for b result in a significant performance/memory improvement to warrant their use.

#### 3.3.2. RHASH

The RHASH is the general term used throughout [RFC7401] to refer to the hash used for a specific HIT suite. For this addendum SHAKE128 is used, even for HIs of EdDSA448.

Unless otherwise specified, L of SHAKE128 is 256, resulting in a similar output to SHA256. Any truncation used for, older, fixed

output hashes is still used. This is to simplify code integration. One exception to this is in Section 4.

#### 3.3.3. HIP_MAC and HIP_MAC2

The HIP_MAC and HIP_MAC2 parameters in [RFC7401] use HMAC [RFC2104]. This performs two hashes on a string with a key for a keyed hash the length of the underlying hash.

Here, KMAC from NIST SP 800-185 [NIST.SP.800-185] is used. This is a single pass using the underlying cSHAKE function. The function call is:

```
KMAC128(Key, Input String, 256, "")
```

### 3.4. HIP Cipher

HIP encrypted parameters use the HIP_CIPHER, Section 5.2.8 of [RFC7401]. The Keccak Keyak cipher, [Keyak_Cipher], is recommended. Keyak is a candidate in the NIST Lightweight Cryptography competition and is consistent with the overall approach in this addendum to use Keccak functions for simplicity in design and implementation.

#### 3.4.1. HIP_CIPHER

The HIP_CIPHER parameter values for Keyak are:

hip_cipher

| Suite ID | Value | |
|---|---|---|
| RIVER KEYAK | 6 | (Keyak) |
| LAKE KEYAK | 7 | (Keyak) |

For use as the HIP Cipher, the TAG generated in Keyak is length 0. The Keyak SUV is the key plus IV specified for the encrypted parameter. River Keyak MAY be used for [Keyak_Cipher], in place of AES-CTR.

Lake Keyak can provide 256 bits of security by following the recommendations for the Keyak cipher.

## 4. Generating a HIT from an HI

The EdDSA/cSHAKE based HITs vary slightly the ORCHID generation method described in section 3.2 of [RFC7401]. The XOF functionality

of cSHAKE produces an output of L bits. This replaces the Encode_96 function in the ORCHID generation.

For identities that are EdDSA public keys, ORCHIDs will be generated per the process defined in Using cSHAKE in ORCHIDs [I-D.moskowitz-orchid-cshake]

## 5. HIP KEYMAT Generation

The KMAC function provides a new, more efficient, key derivation function over HKDF [RFC5869]. This will be referred to as KKDF.

The choice of KMAC128 or KMAC256 is based on the strength of the output key material. For 256 bits of strength equivalent to HMAC-SHA256, use KMAC256. Per [NIST.SP.800-56Cr1], Section 4.1, Option 3:

```
OKM = KMAC[128|256](salt | info, IKM, L, S)
```

L is the derived key bit length. Since 4 HIP keys are "drawn" from this output, the length is 4 * HIP_key_size. Per ASIACRYPT 2017, pp. 606-637 [ASIACRYPT-2017] each of these derived keys will have the same strength as the Diffie-Hellman shared secret.

S is the byte string 01001011 || 01000100 || 01000110, which represents the sequence of characters "K", "D", and "F" in 8-bit ASCII.

Salt and info are derived as defined in [RFC7401] or [I-D.ietf-hip-dex]. There are special security considerations for IKM per [RFC7748]. The two HIs MUST be used in constructing IKM as follows:

```
IKM = Diffie-Hellman secret | HI-R |  HI-I
```

These are separately DER encoded.

## 6. Using Keccak for a Pseudorandom Function

Appendix B of NIST SP 800-185 [NIST.SP.800-185] defines how to use SHAKE, cSHAKE, or KMAC as a PRF.

## 7. IANA Considerations

IANA will need to make the following changes to the "Host Identity Protocol (HIP) Parameters" registries:

**Diffie Hellman:**

This document defines the new Curve25519 and Curve448 for the Diffie-Hellman exchange (see Section 3.1.1).

**Host ID:**
This document defines the new EdDSA Host ID (see Section 3.2.1).

**HIT Suite ID:**
This document defines the new HIT Suite of EdDSA/cSHAKE (see Section 3.2.2).

**HIP Cipher:**
This document defines the new Keyak ciphers for HIP encrypted parameters (see Section 3.4.1).

## 8. Security Considerations

### 8.1. Keymat vulnerabilities

[RFC7748] warns about using Curve25519 and Curve448 in Diffie-Hellman for key derivation:

Designers using these curves should be aware that for each public key, there are several publicly computable public keys that are equivalent to it, i.e., they produce the same shared secrets. Thus using a public key as an identifier and knowledge of a shared secret as proof of ownership (without including the public keys in the key derivation) might lead to subtle vulnerabilities.

This applies to [I-D.ietf-hip-dex], but may have broader consequences. Thus the two Host IDs are included with the Diffie-Hellman secret.

### 8.2. KMAC Security as a KDF

Section 4.1 of NIST SP 800-185 [NIST.SP.800-185] states:

"The KECCAK Message Authentication Code (KMAC) algorithm is a PRF and keyed hash function based on KECCAK . It provides variable-length output"

That is, the output of KMAC is indistinguishable from a random string, regardless of the length of the output. As such, the output of KMAC can be divided into multiple substrings, each with the strength of the function (KMAC128 or KMAC256) and provided that a long enough key is used, as discussed in Sec. 8.4.1 of SP 800-185.

For example KMAC128(K, X, 512, S), where K is at least 128 bits, can produce 4 128 bit keys each with a strength of 128 bits. That is a single sponge operation is replacing perhaps 5 HMAC-SHA256 operations (each 2 SHA256 operations) in HKDF.

## 9. Acknowledgments

Quynh Dang of NIST gave considerable guidance on using Keccak and the NIST supporting documents. Joan Deamen of the Keccak team was especially helpful in many aspects of using Keccak, particularly with the KEYMAT section and the strength of the derived keys.

## 10. References

### 10.1. Normative References

**[NIST.FIPS.202]** Dworkin, M., "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions", National Institute of Standards and Technology report, DOI 10.6028/nist.fips.202, July 2015, <https://doi.org/10.6028/nist.fips.202>.

**[NIST.SP.800-185]** Kelsey, J., Change, S., and R. Perlner, "SHA-3 derived functions: cSHAKE, KMAC, TupleHash and ParallelHash", National Institute of Standards and Technology report, DOI 10.6028/nist.sp.800-185, December 2016, <https://doi.org/10.6028/nist.sp.800-185>.

**[NIST.SP.800-56Cr1]** Barker, E., Chen, L., and R. Davis, "Recommendation for key-derivation methods in key-establishment schemes", National Institute of Standards and Technology report, DOI 10.6028/nist.sp.800-56cr1, April 2018, <https://doi.org/10.6028/nist.sp.800-56cr1>.

**[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>.

**[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>.

### 10.2. Informative References

**[ASIACRYPT-2017]** Daemen, J., Mennink, B., and G. Van Assche, "Full-State Keyed Duplex with Built-In Multi-user Support", DOI 10.1007/978-3-319-70697-9_21, Advances in Cryptology - ASIACRYPT 2017 pp. 606-637, 2017, <https://doi.org/10.1007/978-3-319-70697-9_21>.

**[I-D.ietf-hip-dex]** Moskowitz, R., Hummen, R., and M. Komu, "HIP Diet EXchange (DEX)", Work in Progress, Internet-Draft, draft-ietf-hip-dex-21, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-hip-dex-21>.

**[I-D.moskowitz-orchid-cshake]**
Moskowitz, R., Card, S., and A. Wiethuechter, "Using cSHAKE in ORCHIDs", Work in Progress, Internet-Draft, draft-moskowitz-orchid-cshake-01, 21 May 2020, <https://tools.ietf.org/html/draft-moskowitz-orchid-cshake-01>.

**[Keccak]** Bertoni, G., Daemen, J., Peeters, M., Van Assche, G., and R. Van Keer, "The Keccak Function", , <https://keccak.team/index.html>.

**[Keyak_Cipher]** Bertoni, G., Daemen, J., Peeters, M., Van Assche, G., and R. Van Keer, "The Keyak Cipher", , <https://keccak.team/keyak.html>.

**[RFC2104]** Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/info/rfc2104>.

**[RFC5869]** Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <https://www.rfc-editor.org/info/rfc5869>.

**[RFC7401]** Moskowitz, R., Ed., Heer, T., Jokela, P., and T. Henderson, "Host Identity Protocol Version 2 (HIPv2)", RFC 7401, DOI 10.17487/RFC7401, April 2015, <https://www.rfc-editor.org/info/rfc7401>.

**[RFC7748]** Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves for Security", RFC 7748, DOI 10.17487/RFC7748, January 2016, <https://www.rfc-editor.org/info/rfc7748>.

**[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>.

## Authors' Addresses

Robert Moskowitz
HTT Consulting
Oak Park, MI 48237
United States of America

Email: rgm@labs.htt-consult.com

Stuart W. Card
AX Enterprize
4947 Commercial Drive

Yorkville, NY 13495
United States of America

Email: stu.card@axenterprize.com

Adam Wiethuechter
AX Enterprize
4947 Commercial Drive
Yorkville, NY 13495
United States of America

Email: adam.wiethuechter@axenterprize.com