Workgroup: DRIP
Updates: 7401, 7343 (if approved)
Published: 27 October 2020
Intended Status: Standards Track
Expires: 30 April 2021
Authors: R. Moskowitz
HTT Consulting
A. Wiethuechter
AX Enterprize, LLC
S. Card
AX Enterprize, LLC
A. Gurtov
Linköping University

UAS Remote ID

Abstract

This document describes the use of Hierarchical Host Identity Tags (HHITs) as self-asserting IPv6 addresses and thereby a trustable Identifier for use as the UAS Remote ID. HHITs include explicit hierarchy to provide Registrar discovery for 3rd-party ID attestation.

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 30 April 2021.

Table of Contents

1. Introduction

[drip-requirements] describes a UAS ID as a "unique (ID-4), non-spoofable (ID-5), and identify a registry where the ID is listed (ID-2)"; all within a 20 character Identifier (ID-1).

This document describes the use of Hierarchical HITs (HHITs) (Appendix B) as self-asserting IPv6 addresses and thereby a trustable Identifier for use as the UAS Remote ID. HHITs include explicit hierarchy to provide Registrar discovery for 3rd-party ID attestation.

HITs are statistically unique through the cryptographic hash feature of second-preimage resistance. The cryptographically-bound addition of the Hierarchy and a HHIT registration process (TBD; e.g. based on Extensible Provisioning Protocol, [RFC5730]) provide complete, global HHIT uniqueness. This is in contrast to general IDs (e.g. a UUID or device serial number) as the subject in an X.509 certificate.

In a multi-CA PKI, a subject can occur in multiple CAs, possibly fraudulently. CAs within the PKI would need to implement an approach to enforce assurance of uniqueness.

Hierarchical HITs are valid, though non-routable, IPv6 addresses. As such, they fit in many ways within various IETF technologies.

1.1. Nontransferablity of HHITs

HIs and its HHITs SHOULD NOT be transferable between UA or even between replacement electronics for a UA. The private key for the HI SHOULD be held in a cryptographically secure component.

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. Notation

| Signifies concatenation of information - e.g., X | Y is the concatenation of X and Y.

2.3. Definitions

See [drip-requirements] for common DRIP terms.

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

HDA (Hierarchical HIT Domain Authority):
The 16 bit field identifying the HHIT Domain Authority under an RAA.

HHIT
Hierarchical Host Identity Tag. A HIT with extra hierarchical information not found in a standard HIT.

HI
Host Identity. The public key portion of an asymmetric keypair used in HIP.

HID (Hierarchy ID):
The 32 bit field providing the HIT Hierarchy ID.

HIP
Host Identity Protocol. The origin of HI, HIT, and HHIT, required for DRIP. Optional full use of HIP enables additional DRIP functionality.

HIT
Host Identity Tag. A 128 bit handle on the HI. HITs are valid IPv6 addresses.

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.

RAA (Registered Assigning Authority):
The 16 bit field identifying the business or organization that manages a registry of HDAs.

RVS (Rendezvous Server):
The HIP Rendezvous Server for enabling mobility, as defined in [RFC8004].

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

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

3. Hierarchical HITs as Remote ID

Hierarchical HITs are a refinement on the Host Identity Tag (HIT) of HIPv2 [RFC7401]. HHITs require a new ORCHID mechanism as described in Appendix C. HHITs for UAS ID also use the new EdDSA/SHAKE128 HIT suite defined in Appendix D (requirements GEN-2). This hierarchy, cryptographically embedded within the HHIT, provides the information for finding the UA's HHIT registry (ID-3).

The current ASTM [F3411-19] specifies three UAS ID types:

TYPE-1 A static, manufacturer assigned, hardware serial number per ANSI/CTA-2063-A "Small Unmanned Aerial System Serial Numbers" [CTA2063A].

TYPE-2 A CAA assigned (presumably static) ID.

TYPE-3 A UTM system assigned UUID [RFC4122], which can but need not be dynamic.

For HHITs to be used effectively as UAS IDs, F3411-19 SHOULD add UAS ID type 4 as HHIT.

3.1. Remote ID as one class of Hierarchical HITs

UAS Remote ID may be one of a number of uses of HHITs. As such these follow-on uses need to be considered in allocating the RAAs Appendix B.3.1 or HHIT prefix assignments Section 8.

3.2. Hierarchy in ORCHID Generation

ORCHIDS, as defined in [RFC7343], do not cryptographically bind the IPv6 prefix nor the Orchid Generation Algorithm (OGA) ID (the HIT Suite ID) to the hash of the HI. The justification then was attacks against these fields are DoS attacks against protocols using them.

HHITs, as defined in Appendix C, cryptographically bind all content in the ORCHID through the hashing function. Thus a recipient of a HHIT that has the underlying HI can directly act on all content in the HHIT. This is especially important to using the hierarchy to find the HHIT Registry.

3.3. Hierarchical HIT Registry

HHITs are registered to Hierarchical HIT Domain Authorities (HDAs). A registration process (TBD) ensures UAS ID global uniqueness (ID-4). It also provides the mechanism to create UAS Public/Private data associated with the HHIT UAS ID (REG-1 and REG-2).

The 2 levels of hierarchy within the HHIT allows for CAAs to have their own Registered Assigning Authority (RAA) for their National Air Space (NAS). Within the RAA, the CAAs can delegate HDAs as needed. There may be other RAAs allowed to operate within a given NAS; this is a policy decision by the CAA.

3.4. Remote ID Authentication using HHITs

The EdDSA25519 Host Identity (HI) [Appendix D] underlying the HHIT can be used in an 84 byte self proof claim as shown in Appendix E to provide proof of Remote ID ownership (requirements GEN-1). An Internet lookup service like DNS can provide the HI and registration proof (requirements GEN-3).

Similarly the 200 byte offline self claim shown in Appendix E.1 provide the same proofs without Internet access and with a small cache that contains the HDA's HI/HHIT and HDA meta-data. These self claims are carried in the ASTM Authentication Message (Msg Type 0x2).

Hashes of previously sent ASTM messages can be placed in a signed "Manifest" Authentication Message (requirements GEN-2). This can be either a standalone Authentication Message, or an enhanced self claim Authentication Message. Alternatively the ASTM Message Pack (Msg Type 0xF) can provide this feature, but only over Bluetooth 5 or WiFi NAN broadcasts.

4. UAS ID HHIT in DNS

There are 2 approaches for storing and retrieving the HHIT from DNS. These are:

* As FQDNs in the .aero TLD.

* Reverse DNS lookups as IPv6 addresses per [RFC8005].

The HHIT can be used to construct an FQDN that points to the USS that has the Public/Private information for the UA (REG-1 and REG-2). For example the USS for the HHIT could be found via the following. Assume that the RAA is 100 and the HDA is 50. The PTR record is constructed as:

```
100.50.hhit.uas.aero   IN PTR      foo.uss.aero.
```

The individual HHITs are potentially too numerous (e.g. 60 - 600M) and dynamic to actually store in a signed, DNS zone. Rather the USS would provide the HHIT detail response.

The HHIT reverse lookup can be a standard IPv6 reverse look up, or it can leverage off the HHIT structure. Assume that the RAA is 10 and the HDA is 20 and the HHIT is:

```
2001:14:28:14:a3ad:1952:ad0:a69e
```

An HHIT reverse lookup would be to is:

```
a69e.ad0.1952.a3ad.14.28.14.2001.20.10.hhit.arpa.
```

5. Other UTM uses of HHITs

HHITs can be used extensively within the UTM architecture beyond UA ID (and USS in UA ID registration and authentication). This includes a GCS HHIT ID. It could use this if it is the source of Network Remote ID for securing the transport and for secure C2 transport [drip-secure-nrid-c2].

Observers SHOULD have HHITs to facilitate UAS information retrieval (e.g., for authorization to private UAS data). They could also use their HHIT for establishing a HIP connection with the UA Pilot for direct communications per authorization. Further, they can be used by FINDER observers, [crowd-sourced-rid].

6. DRIP Requirements addressed

This document provides solutions to GEN 1 - 3, ID 1 - 5, and REG 1 - 2.

7. ASTM Considerations

ASTM will need to make the following changes to the "UA ID" in the Basic Message (Msg Type 0x0):

Type 4:
This document UA ID of Hierarchical HITs (see Section 3).

8. IANA Considerations

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

Host ID:
This document defines the new EdDSA Host ID (see Appendix D.1).

HIT Suite ID:
This document defines the new HIT Suite of EdDSA/cSHAKE (see Appendix D.2).

HIT Suite ID:
This document defines two new HDA domain HIT Suites (see Appendix B.2.1).

Because HHIT format is not compatible with [RFC7343], IANA is requested to allocated a new 28-bit prefix out of the IANA IPv6 Special Purpose Address Block, namely 2001:0000::/23, as per [RFC6890].

9. Security Considerations

A 64 bit hash space presents a real risk of second pre-image attacks Section 9.2. The HHIT Registry services effectively block attempts to "take over" a HHIT. It does not stop a rogue attempting to impersonate a known HHIT. This attack can be mitigated by the receiver of the HHIT using DNS to find the HI for the HHIT.

Another mitigation of HHIT hijacking is if the HI owner (UA) supplies an object containing the HHIT and signed by the HI private key of the HDA such as Appendix E.1 as shown in Section 3.4.

The two risks with hierarchical HITs are the use of an invalid HID and forced HIT collisions. The use of a DNS zone (e.g. "hhit.arpa.") is a strong protection against invalid HIDs. Querying an HDA's RVS for a HIT under the HDA protects against talking to unregistered clients. The Registry service has direct protection against forced or accidental HIT hash collisions.

Cryptographically Generated Addresses (CGAs) provide a unique assurance of uniqueness. This is two-fold. The address (in this case the UAS ID) is a hash of a public key and a Registry hierarchy naming. Collision resistance (more important that it implied second-preimage resistance) makes it statistically challenging to attacks. A registration process (TBD) within the HDA provides a level of assured uniqueness unattainable without mirroring this approach.

The second aspect of assured uniqueness is the digital signing process of the HHIT by the HI private key and the further signing of the HI public key by the Registry's key. This completes the ownership process. The observer at this point does not know WHAT owns the HHIT, but is assured, other than the risk of theft of the HI private key, that this UAS ID is owned by something and is properly registered.

9.1. Hierarchical HIT Trust

The HHIT UAS RID in the ASTM Basic Message (Msg Type 0x0, the actual Remote ID message) does not provide any assertion of trust. The best that might be done within this Basic Message is 4 bytes truncated from a HI signing of the HHIT (the UA ID field is 20 bytes and a HHIT is 16). This is not trustable. Minimally, it takes 84 bytes, Appendix E, to prove ownership of a HHIT.

The ASTM Authentication Messages (Msg Type 0x2) as shown in Section 3.4 can provide practical actual ownership proofs. These claims include timestamps to defend against replay attacks. But in themselves, they do not prove which UA actually sent the message. They could have been sent by a dog running down the street with a Broadcast Remote ID device strapped to its back.

Proof of UA transmission comes when the Authentication Message includes proofs for the ASTM Location/Vector Message (Msg Type 0x1) and the observer can see the UA or that information is validated by ground multilateration [crowd-sourced-rid]. Only then does an observer gain full trust in the HHIT Remote ID.

HHIT Remote IDs obtained via the Network Remote ID path provides a different approach to trust. Here the UAS SHOULD be securely communicating to the USS (see [drip-secure-nrid-c2]), thus asserting HHIT RID trust.

9.2. Collision risks with Hierarchical HITs

The 64 bit hash size does have an increased risk of collisions over the 96 bit hash size used for the other HIT Suites. There is a 0.01% probability of a collision in a population of 66 million. The probability goes up to 1% for a population of 663 million. See Appendix F for the collision probability formula.

However, this risk of collision is within a single "Additional Information" value, i.e. a RAA/HDA domain. The UAS/USS registration process should include registering the HHIT and MUST reject a collision, forcing the UAS to generate a new HI and thus HHIT and reapplying to the registration process.

10. References

10.1. Normative References

[F3411-19] ASTM International, "Standard Specification for Remote ID and Tracking", February 2020, <http://www.astm.org/cgi-bin/resolver.cgi?F3411>.

[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>.

[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>.

[RFC6890] Cotton, M., Vegoda, L., Bonica, R., Ed., and B. Haberman, "Special-Purpose IP Address Registries", BCP 153, RFC 6890, DOI 10.17487/RFC6890, April 2013, <https://www.rfc-editor.org/info/rfc6890>.

[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>.

[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

[corus] CORUS, "U-space Concept of Operations", September 2019, <https://www.sesarju.eu/node/3411>.

[crowd-sourced-rid] Moskowitz, R., Card, S., Wiethuechter, A., Zhao, S., and H. Birkholz, "Crowd Sourced Remote ID", Work in Progress, Internet-Draft, draft-moskowitz-drip-crowd-sourced-rid-04, 20 May 2020, <https://tools.ietf.org/html/draft-moskowitz-drip-crowd-sourced-rid-04>.

[CTA2063A] ANSI, "Small Unmanned Aerial Systems Serial Numbers", September 2019.

[drip-requirements] Card, S., Wiethuechter, A., Moskowitz, R., and A. Gurtov, "Drone Remote Identification Protocol (DRIP)

Requirements", Work in Progress, Internet-Draft, draft-ietf-drip-reqs-04, 25 August 2020, <https://tools.ietf.org/html/draft-ietf-drip-reqs-04>.

[drip-secure-nrid-c2]
Moskowitz, R., Card, S., Wiethuechter, A., and A. Gurtov, "Secure UAS Network RID and C2 Transport", Work in Progress, Internet-Draft, draft-moskowitz-drip-secure-nrid-c2-01, 27 September 2020, <https://tools.ietf.org/html/draft-moskowitz-drip-secure-nrid-c2-01>.

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

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <https://www.rfc-editor.org/info/rfc4122>.

[RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.17487/RFC5730, August 2009, <https://www.rfc-editor.org/info/rfc5730>.

[RFC7343] Laganier, J. and F. Dupont, "An IPv6 Prefix for Overlay Routable Cryptographic Hash Identifiers Version 2 (ORCHIDv2)", RFC 7343, DOI 10.17487/RFC7343, September 2014, <https://www.rfc-editor.org/info/rfc7343>.

[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>.

[RFC8004] Laganier, J. and L. Eggert, "Host Identity Protocol (HIP) Rendezvous Extension", RFC 8004, DOI 10.17487/RFC8004, October 2016, <https://www.rfc-editor.org/info/rfc8004>.

[RFC8005] Laganier, J., "Host Identity Protocol (HIP) Domain Name System (DNS) Extension", RFC 8005, DOI 10.17487/RFC8005, October 2016, <https://www.rfc-editor.org/info/rfc8005>.

Appendix A. EU U-Space RID Privacy Considerations

EU is defining a future of airspace management known as U-space within the Single European Sky ATM Research (SESAR) undertaking. Concept of Operation for EuROpean UTM Systems (CORUS) project proposed low-level Concept of Operations [corus] for UAS in EU. It introduces strong requirements for UAS privacy based on European GDPR regulations. It suggests that UAs are identified with agnostic

IDs, with no information about UA type, the operators or flight trajectory. Only authorized persons should be able to query the details of the flight with a record of access.

Due to the high privacy requirements, a casual observer can only query U-space if it is aware of a UA seen in a certain area. A general observer can use a public U-space portal to query UA details based on the UA transmitted "Remote identification" signal. Direct remote identification (DRID) is based on a signal transmitted by the UA directly. Network remote identification (NRID) is only possible for UAs being tracked by U-Space and is based on the matching the current UA position to one of the tracks.

The project lists "E-Identification" and "E-Registrations" services as to be developed. These services can follow the privacy mechanism proposed in this document. If an "agnostic ID" above refers to a completely random identifier, it creates a problem with identity resolution and detection of misuse. On the other hand, a classical HIT has a flat structure which makes its resolution difficult. The Hierarchical HITs provide a balanced solution by associating a registry with the UA identifier. This is not likely to cause a major conflict with U-space privacy requirements, as the registries are typically few at a country level (e.g. civil personal, military, law enforcement, or commercial).

Appendix B. The Hierarchical Host Identity Tag (HHIT)

The Hierarchical HIT (HHIT) is a small but important enhancement over the flat HIT space. By adding two levels of hierarchical administration control, the HHIT provides for device registration/ ownership, thereby enhancing the trust framework for HITs.

HHITs represent the HI in only a 64 bit hash and uses the other 32 bits to create a hierarchical administration organization for HIT domains. Hierarchical HIT construction is defined in Appendix C. The input values for the Encoding rules are in Appendix C.1.

A HHIT is built from the following fields:

*IANA prefix (max 28 bit)

*32 bit Hierarchy ID (HID)

*4 (or 8) bit HIT Suite ID

*ORCHID hash (96 - prefix length - Suite ID length bits, e.g. 64) See Appendix C

B.1. HHIT prefix

A unique IANA IPv6 prefix, no larger than 28 bit, for HHITs is recommended. It clearly separates the flat-space HIT processing from HHIT processing per Appendix C.

Without a unique prefix, the first 4 bits of the RRA would be interpreted as the HIT Suite ID per HIPv2 [RFC7401].

B.2. HHIT Suite IDs

The HIT Suite IDs specifies the HI and hash algorithms. Any HIT Suite ID can be used for HHITs. The 8 bit format is supported (only when the first 4 bits are ZERO), but this reduces the ORCHID hash length.

B.2.1. 8 bit HIT Suite IDs

Support for 8 bit HIT Suite IDs is allowed in Sec 5.2.10, [RFC7401], but not specified in how ORCHIDs are generated with these longer OGAs. Appendix C provides the algorithmic flexiblity, allowing for HDA custom HIT Suite IDs as follows:

HIT Suite	Four-bit ID	Eight-bit encoding
HDA Assigned 1	NA	0x0E
HDA Assigned 2	NA	0x0F

This feature may be used for large-scale experimenting with post quantum computing hashes or similar domain specific needs. Note that currently there is no support for domain specific HI algorithms.

B.3. The Hierarchy ID (HID)

The Hierarchy ID (HID) provides the structure to organize HITs into administrative domains. HIDs are further divided into 2 fields:

*16 bit Registered Assigning Authority (RAA)

*16 bit Hierarchical HIT Domain Authority (HDA)

B.3.1. The Registered Assigning Authority (RAA)

An RAA is a business or organization that manages a registry of HDAs. For example, the Federal Aviation Authority (FAA) could be an RAA.

The RAA is a 16 bit field (65,536 RAAs) assigned by a numbers management organization, perhaps ICANN's IANA service. An RAA must

provide a set of services to allocate HDAs to organizations. It must have a public policy on what is necessary to obtain an HDA. The RAA need not maintain any HIP related services. It must maintain a DNS zone minimally for discovering HID RVS servers.

As HHITs may be used in many different domains, RAA should be allocated in blocks with consideration on the likely size of a particular usage. Alternatively, different Prefixes can be used to separate different domains of use of HHTs.

This DNS zone may be a PTR for its RAA. It may be a zone in a HHIT specific DNS zone. Assume that the RAA is 100. The PTR record could be constructed:

```
100.hhit.arpa   IN PTR      raa.bar.com.
```

B.3.2. The Hierarchical HIT Domain Authority (HDA)

An HDA may be an ISP or any third party that takes on the business to provide RVS and other needed services for HIP enabled devices.

The HDA is an 16 bit field (65,536 HDAs per RAA) assigned by an RAA. An HDA should maintain a set of RVS servers that its client HIP-enabled customers use. How this is done and scales to the potentially millions of customers is outside the scope of this document. This service should be discoverable through the DNS zone maintained by the HDA's RAA.

An RAA may assign a block of values to an individual organization. This is completely up to the individual RAA's published policy for delegation.

Appendix C. ORCHIDs for Hierarchical HITs

This section improves on ORCHIDv2 [RFC7343] with three enhancements:

- Optional Info field between the Prefix and OGA ID.
- Increased flexibility on the length of each component in the ORCHID construction, provided the resulting ORCHID is 128 bits.
- Use of cSHAKE, NIST SP 800-185 [NIST.SP.800-185], for the hashing function.

The [Keccak] based cSHAKE XOF hash function is a variable output length hash function. As such it does not use the truncation operation that other hashes need. The invocation of cSHAKE specifies the desired number of bits in the hash output. Further, cSHAKE has a

parameter 'S' as a customization bit string. This parameter will be used for including the ORCHID Context Identifier in a standard fashion.

This ORCHID construction includes the fields in the ORCHID in the hash to protect them against substitution attacks. It also provides for inclusion of additional information, in particular the hierarchical bits of the Hierarchical HIT, in the ORCHID generation. This should be viewed as an addendum to ORCHIDv2 [RFC7343], as it can produce ORCHIDv2 output.

C.1. Adding additional information to the ORCHID

ORCHIDv2 [RFC7343] is currently defined as consisting of three components:

```
ORCHID     :=  Prefix | OGA ID | Encode_96( Hash )

where:

Prefix          : A constant 28-bit-long bitstring value
                  (IANA IPv6 assigned).

OGA ID          : A 4-bit long identifier for the Hash_function
                  in use within the specific usage context.  When
                  used for HIT generation this is the HIT Suite ID.

Encode_96( )    : An extraction function in which output is obtained
                  by extracting the middle 96-bit-long bitstring
                  from the argument bitstring.
```

This addendum will be constructed as follows:

```
ORCHID     :=  Prefix (p) | Info (n) | OGA ID (o) | Hash (m)

where:

Prefix (p)      : An IANA IPv6 assigned prefix (max 28-bit-long).

Info (n)        : n bits of information that define a use of the
                  ORCHID.  n can be zero, that is no additional
                  information.

OGA ID (o)      : A 4 or 8 bit long identifier for the Hash_function
                  in use within the specific usage context.  When
                  used for HIT generation this is the HIT Suite ID.

Hash (m)        : An extraction function in which output is m bits.

p + n + o + m = 128 bits
```

With a 28 bit IPv6 Prefix, the remaining 100 bits can be divided in any manner between the additional information, OGA ID, and the hash output. Care must be taken in determining the size of the hash portion, taking into account risks like pre-image attacks. Thus 64 bits as used in Hierarchical HITs may be as small as is acceptable.

C.2. ORCHID Encoding

This addendum adds a different encoding process to that currently used in ORCHIDv2. The input to the hash function explicitly includes all the header content plus the Context ID. The header content consists of the Prefix, the Additional Information, and OGA ID (HIT Suite ID). Secondly, the length of the resulting hash is set by sum of the length of the ORCHID header fields. For example, a 28 bit Prefix with 32 bits for the HID and 4 bits for the OGA ID leaves 64 bits for the hash length.

To achieve the variable length output in a consistent manner, the cSHAKE hash is used. For this purpose, cSHAKE128 is appropriate. The the cSHAKE function call for this addendum is:

```
cSHAKE128(Input, L, "", Context ID)

Input      :=  Prefix | Additional Information | OGA ID | HOST_ID
L          :=  Length in bits of hash portion of ORCHID
Context ID :=  0x00B5 A69C 795D F5D5 F008 7F56 843F 2C40
```

For full Suite ID support (those that use fixed length hashes like SHA256), the following hashing can be used (Note: this does NOT produce output Identical to ORCHIDv2 for Prefix of /28 and Additional Information of ZERO length):

```
 Hash[L](Context ID | Input)

 Input      :=  Prefix | Additional Information | OGA ID | HOST_ID
 L          :=  Length in bits of hash portion of ORCHID
 Context ID :=  0x00B5 A69C 795D F5D5 F008 7F56 843F 2C40

 Hash[L]    :=  An extraction function in which output is obtained
                by extracting the middle L-bit-long bitstring
                from the argument bitstring.
```

Hierarchical HIT uses the same context as all other HIPv2 HIT Suites as they are clearly separated by the distinct HIT Suite ID.

C.2.1. Encoding ORCHIDs for HITv2

This section is included to provide backwards compatibility for ORCHIDv2 [RFC7343] as used for HITv2 [RFC7401].

For HITv2s, the Prefix MUST be 2001:20::/28. Info is length ZERO (not included), and OGA ID is length 4. Thus the HI Hash is length 96. Further the Prefix and OGA ID are NOT included in the hash calculation. Thus the following ORCHID calculations for fixed output length hashes are used:

```
 Hash[L](Context ID | Input)

 Input      :=  HOST_ID
 L          :=  96
 Context ID :=  0xF0EF F02F BFF4 3D0F E793 0C3C 6E61 74EA

 Hash[L]    :=  An extraction function in which output is obtained
                by extracting the middle L-bit-long bitstring
                from the argument bitstring.
```

For variable output length hashes use:

```
Hash[L](Context ID | Input)

Input      :=  HOST_ID
L          :=  96
Context ID :=  0xF0EF F02F BFF4 3D0F E793 0C3C 6E61 74EA

Hash[L]    :=  The L bit output from the hash function
```

Then the ORCHID is constructed as follows:

```
Prefix | OGA ID | Hash Output
```

C.3. ORCHID Decoding

With this addendum, the decoding of an ORCHID is determined by the Prefix and OGA ID (HIT Suite ID). ORCHIDv2 [RFC7343] decoding is selected when the Prefix is: 2001:20::/28.

For Hierarchical HITs, the decoding is determined by the presence of the HHIT Prefix as specified in the HHIT document.

C.4. Decoding ORCHIDs for HITv2

This section is included to provide backwards compatibility for ORCHIDv2 [RFC7343] as used for HITv2 [RFC7401].

HITv2s are identified by a Prefix of 2001:20::/28. The next 4 bits are the OGA ID. is length 4. The remaining 96 bits are the HI Hash.

Appendix D. Edward Digital Signature Algorithm for HITs

Edwards-Curve Digital Signature Algorithm (EdDSA) [RFC8032] are specified here for use as Host Identities (HIs) per HIPv2 [RFC7401]. Further the HIT_SUITE_LIST is specified as used in [RFC7343].

See Appendix B.2 for use of the HIT Suite for this document.

D.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]

D.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 this Appendix.

The output of cSHAKE128 is variable per the needs of a specific ORCHID construction. It is at most 96 bits long and is directly used in the ORCHID (without truncation).

Index	Hash function	HMAC	Signature algorithm family	Description
5	cSHAKE128	KMAC128	EdDSA	EdDSA HI hashed with cSHAKE128, output is variable

Table 1: HIT Suites

Appendix E. Example HHIT Self Claim

This section shows example uses of HHIT RID to prove trustworthiness of the RID. These are examples only and other documents will provide fully specified claims. Care has been taken in the example design to minimize the risk of replay attacks.

Ownership of a HHIT can be proved in 84 bytes via the following HHIT Self Claim:

* 4 byte Signing Timestamp

* 16 byte HHIT

* 64 byte Signature (EdDSA25519 signature)

The Timestamp MAY be the standard UNIX time at the time of signing. A protocol specific timestamp may be used to avoid programming complexities. For example, [F3411-19] uses a 00:00:00 01/01/2019 offset.

To minimize the risk of replay, the UA SHOULD create a new Self Claim, with a new timestamp, at least once a minute. The UA MAY precompute these claims and transmit during the appropriate 1 minute window. 1 minute is chosen as a balance between claim compute time against risk. A shorter window of use lessens the risk of replay.

The signature is over the 20 byte Timestamp + HHIT.

The receiver of such a claim would need access to the underlying public key (HI) to validate the signature. This may be obtained via a DNS query using the HHIT. A larger (116 bytes) Self Claim could include the EdDSA25519 HI.

E.1. HHIT Offline Self Claim

Ownership of a HHIT can be proved in 200 bytes without Internet access and a small cache via the following HHIT Offline Self Claim:

* 16 byte UA HHIT

* 32 byte UA EdDSA25519 HI

*4 byte HDA Signing Expiry Timestamp

*16 byte HDA HHIT

*64 byte HDA Signature (EdDSA25519 signature)

*4 byte UA Signing Timestamp

*64 byte UA Signature (EdDSA25519 signature)

The Timestamps MAY be the standard UNIX time at the time of signing. A protocol specific timestamp may be used to avoid programming complexities. For example, [F3411-19] uses a 00:00:00 01/01/2019 offset.

The HDA signature is over the 68 byte UA HHIT + UA HI + HDA Expiry Timestamp + HDA HHIT. During the UA Registration process, the UA would provide a Self Claim to the HDA. The HDA would construct its claim of registry with an Expiry Timestamp, its own HHIT, and its signature, returning a 132 byte HDA Registry Claim to the UA. The UA would use this much the same way as its HHIT only in the Self Claim above, creating a 200 byte Offline Self Claim.

The receiver of such a claim would need a cache of RAA ID, HDA ID, HDA HHIT, and HDA HI (min 80 bytes per RAA/HDA).

Appendix F. Calculating Collision Probabilities

The accepted formula for calculating the probability of a collision is:

$$p = 1 - e^{-k^2/(2n)}$$

P Collision Probability
n Total possible population
k Actual population

The following table provides the approximate population size for a collision for a given total population.

Total Population	Deployed Population With Collision Risk of .01%	1%
2^96	4T	42T
2^72	1B	10B
2^68	250M	2.5B
2^64	66M	663M
2^60	16M	160M

Acknowledgments

Dr. Gurtov is an adviser on Cybersecurity to the Swedish Civil Aviation Administration.

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.

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, LLC
4947 Commercial Drive
Yorkville, NY 13495
United States of America

Email: stu.card@axenterprize.com

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

Email: adam.wiethuechter@axenterprize.com

Andrei Gurtov
Linköping University
IDA
SE-58183 Linköping

Sweden

Email: gurtov@acm.org