DRIP
Internet-Draft
Updates: 7401, 7343 (if approved)
Intended status: Standards Track
Expires: 8 May 2022

R. Moskowitz
HTT Consulting
S. Card
A. Wiethuechter
AX Enterprize, LLC
A. Gurtov
Linköping University
4 November 2021

DRIP Entity Tag (DET) for Unmanned Aircraft System Remote Identification (UAS RID)
draft-ietf-drip-rid-13

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 Unmanned Aircraft System Remote Identification and tracking (UAS RID). Within the context of RID, HHITs will be called DRIP Entity Tags (DET). HHITs self-attest to the included explicit hierarchy that provides Registrar discovery for 3rd-party identifier 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 8 May 2022.

Commenté [BMI1]: Those are listed as informative. I guess the updating requires them to be listed as normative.

Table of Contents

1. Introduction

[drip-requirements] describes an Unmanned Aircraft System Remote Identification and tracking (UAS ID) as 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 Host Identity Tags (HHITs) (Section 3) as self-asserting IPv6 addresses and thereby a trustable identifier for use as the UAS Remote ID. HHITs include explicit hierarchy to enable DNS HHIT queries (Host ID for authentication, e.g., [drip-authentication]) and for Extensible Provisioning Protocol (EPP) Registrar discovery [RFC7484] for 3rd-party identification attestation (e.g., [drip-authentication]).

HHITs as used within the context of UAS will be labeled as DRIP Entity Tags (DET). Throughout this document HHIT and DET will be used appropriately. HHIT will be used when ~~convering~~covering the technology, and DET for their context within UAS RID.

HHITs are statistically unique through the cryptographic hash feature of second-preimage resistance. The cryptographically-bound addition of the ~~Hierarchy~~ hierarchy and a HHIT registration process [drip-registries] provide complete, global HHIT uniqueness. This ~~is in~~ contrasts ~~to~~with using general identifiers (e.g., ~~a~~ Universally Unique IDentifiers (UUIDs) [RFC4122] or device serial numbers) as the subject in an X.509 [RFC5280] certificate.

In a ~~multi-CA (~~multi Certificate Authority (multi-CA) PKI alternative to HHITs, a Remote ID as the Subject (Section 4.1.2.6 of [RFC5280]) can occur in multiple CAs, possibly fraudulently. CAs within the PKI would need to implement an approach to enforce assurance of the uniqueness achieved with HHITs.

Hierarchical HITs provide self-attestation of the HHIT registry. A HHIT can only be in a single registry within a registry system (e.g. ~~Extensible Provisioning Protocol (~~EPP~~)~~ ~~[RFC5730]~~ and DNS).

Hierarchical HITs are valid, though non-routable, IPv6 addresses [RFC8200]. As such, they fit in many ways within various IETF technologies.

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. Notations

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

2.3. Definitions

This document uses the terms defined in [drip-requirements]. The following new terms are used in the document:

cSHAKE (The customizable SHAKE function [NIST.SP.800-185]):
: Extends the SHAKE [NIST.FIPS.202] scheme to allow users to customize their use of the SHAKE function.

HDA (Hierarchical HIT Domain Authority):
: The 16-bit field that identifies the HHIT Domain Authority under ~~an~~ a Registered Assigning Authority (RAA).

HHIT
: Hierarchical Host Identity Tag. A HIT with extra hierarchical information not found in a standard HIT [RFC7401].

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

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

HIP (Host Identity Protocol)
: The origin of HI, HIT, and HHIT, required for DRIP.

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

Commenté [BMI2]: As we don't require or assume the use of HIP for DRIP, I suggest to generalize the use of HI.

Mis en forme : Surlignage

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. ~~In particular a~~All the functions referenced from [NIST.FIPS.202] and [NIST.SP.800-185].

KMAC (KECCAK Message Authentication Code [NIST.SP.800-185]):
A PRF and keyed hash function based on KECCAK.

Commenté [BMI3]: Please expand

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

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

Commenté [BMI4]: As HHITs can be useful in drip context even if HIP is not used.

SHAKE (Secure Hash Algorithm KECCAK [NIST.FIPS.202]):
A secure hash that allows for an arbitrary output length.

XOF (eXtendable-Output Function [NIST.FIPS.202]):
A function on bit strings (also called messages) in which the output can be extended to any desired length.

3. 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~~ 64-bit hash and use~~s~~ the other 32 bits to create a hierarchical administration organization for HIT domains. Hierarchical HIT construction is defined in Section 3.5. The input values for the Encoding rules are described in Section 3.5.1.

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 Section 3.5

Commenté [BMI5]: Please expand

The Context ID for the ORCHID hash is:

Context ID := 0x00B5 A69C 795D F5D5 F008 7F56 843F 2C40

A python script is available for generating HHITs [hhit-gen].

3.1. HHIT ~~prefix~~Prefix for RID Purposes

A unique IANA IPv6 prefix, no larger than 28 bits~~bit~~, for HHITs is recommended. It clearly separates the flat-space HIT processing from HHIT processing per Section 3.5.

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

3.2. HHIT Suite IDs

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

3.2.1. ~~8~~ 8-bit HIT Suite IDs

Support for 8~~-~~-bit HIT Suite IDs is allowed in Section 5.2.10 of [RFC7401], but not specified in how ORCHIDs are generated with these longer OGAs. Section 3.5 provides the algorithmic flexibility, allowing for HDA custom HIT Suite IDs as follows:

HIT Suite	Four-bit ID	Eight-bit encoding
HDA Assigned 1	NA	TBD3 (suggested value 0x0E)
HDA Assigned 2	NA	TBD4 (suggested value 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~~ domain-specific HI algorithms.

3.3. The Hierarchy ID (HID)

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

* ~~16~~ 16-bit Registered Assigning Authority (RAA)

* ~~16~~ 16-bit Hierarchical HIT Domain Authority (HDA)

3.3.1. 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.

Commenté [BMI6]: Please expand

Commenté [BMI7]: May be add a pointer to the IANA section

Commenté [BMI8]: I suspect this claim prompt some comments during future reviews. It would helpful if the reasoning is elaborated here.

The RAA is a ~~16~~ 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~~ 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 an HHIT specific DNS zone. Assume that the RAA is 100. The PTR record could be constructed as follows:

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

3.3.2. 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~~ such as those required for ~~HIP~~ HIP-enabled devices.

The HDA is ~~an~~ a ~~16~~ 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~~ are 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. Such policy is out of scope.

3.4. 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]. Furthermore, the HIT_SUITE_LIST is specified as used in [RFC7343].

See Section 3.2 for use of the HIT Suite in the context of ~~for~~ this document.

3.4.1. 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].

Commenté [BMI9]: I suggest we avoid this weak wording, but ask formally IANA to maintain this space.

Some guidelines for RAA assignments should be provided.

For example, reserve a block for IETF assignment, and the large block for Expert Review. A value can be reserved for experimentation.

Commenté [BMI10]: We need to include this clarification in the introduction to further insist on this.

Mis en forme : Surlignage

Commenté [BMI11]: To avoid requiring HIP

Commenté [BMI12]: Why do we need to mention this?

Algorithm profiles	Values	
EdDSA	TBD1 (suggested value 13) [RFC8032]	(RECOMMENDED)

For hosts that implement EdDSA as the algorithm, the following EdDSA 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]

3.4.2. 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].

Commenté [BMI13]: That is in the context of DRIP?

Commenté [BMI14]: Idem as the previous comment

The following HIT Suite ID is defined, and the relationship between the ~~four~~4-bit ID value used in the OGA ID field and the ~~eight~~8-bit encoding within the HIT_SUITE_LIST ID field is clarified:

HIT Suite	4-bit ID	8-bit encoding	
RESERVED	0	0x00	
EdDSA/cSHAKE128	TBD2 (suggested value 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.

Commenté [BMI15]: Which one ?

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	Description
5	cSHAKE128	KMAC128	EdDSA	EdDSA HI hashed with cSHAKE128, output is variable

Table 1: HIT Suites

3.5. 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 [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.

3.5.1. 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~~ 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. Note that if a ~~8~~ 8-bit OGA is used, the hash may be 4 bits shorter. This may result in a greater risk of pre-image attacks and a corresponding greater need to manage HHIT registration and require look up of the HI from a trusted source.

3.5.2. 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~~ 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
```

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

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.

3.5.2.1. Encoding ORCHIDs for HITv2

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

Commenté [BMI16]: There is no such notion in rfc7401.

For HITv2s, the Prefix ~~MUST be~~ is 2001:20::/28. Info is zero-length ~~ZERO~~ (i.e., not included), and OGA ID is length 4. Thus, the HI Hash is length 96. Further the Prefix and OGA ID are ~~NOT~~ 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
```

3.5.3. 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 (Section XX) ~~as specified in the HHIT document~~.

Commenté [BMI17]: Add a pointer to the IANA section where the prefix is requested

3.5.4. 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.

Commenté [BMI18]: Please check

4. Hierarchical HITs as Remote ID DRIP Entity Tags (DET)

Hierarchical HITs are a refinement on the Host Identity Tag (HIT) of HIPv2 ~~[RFC7401]~~. HHITs require a new ~~Overlay Routable Cryptographic Hash Identifier (~~ORCHID ~~[RFC7343])~~ mechanism as described in Section 3.5.

HHITs for UAS ID (called, DETs) also use the new EdDSA/SHAKE128 HIT suite defined in Section 3.4 (GEN-2 in [drip-requirements]). This hierarchy, cryptographically embedded within the HHIT, provides the information for finding the UA's HHIT registry (ID-3 in [drip-requirements]).

ASTM Standard Specification for Remote ID and Tracking [F3411] specifies four 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]. These can be dynamic, but do not need to be.

TYPE-4 Specific Session ID (SSI)

Note that Types 1 - 3 allow for an UAS ID with a maximum length of 20 bytes, the SSI (Type 4) uses the first byte of the ID for the SSI value, thus restricting the UAS ID to a maximum of 19 bytes. The SSI values initially assigned (as per 2021) are:

ID 1 IETF - DRIP Drone Remote Identification Protocol (DRIP) entity ID.

ID 2 3GPP - IEEE 1609.2-2016 HashedID8

4.1. Nontransferablity of ~~HHITs~~DETs

A HI and its HHIT SHOULD NOT be transferable between UA or even between replacement electronics (e.g., replacement of damaged controller CPU) for a UA. The private key for the HI SHOULD be held in a cryptographically secure component.

4.2. Encoding HHITs in CTA 2063-A Serial Numbers

In some ~~cases~~cases, it is advantageous to encode HHITs as a CTA 2063-A Serial Number [CTA2063A]. For example, the FAA Remote ID Rules [FAA_RID] state that a Remote ID Module (i.e.~~,~~ not integrated with UA controller) must only use "the serial number of the unmanned aircraft"; CTA 2063-A meets this requirement.

Encoding a~~n~~ HHIT within the CTA 2063-A format is not simple. The CTA 2063-A format is defined as:

Serial Number := MFR Code | Length Code | MFR SN

where:

MFR Code : 4 character code assigned by ICAO.

Length Code : 1 character Hex encoding of MFR SN length (1-F).

MFR SN : Alphanumeric code (0-9, A-Z except O and I). Maximum length of 15 characters.

There is no place for the HID; there will need to be a mapping service from Manufacturer Code to HID. The HIT Suite ID and ORCHID hash will take 14 characters (see below), leaving 1 character to distinguish encoded DETs from other manufacturer use of CTA 2063-A Serial Numbers.

Commenté [BMI19]: Can be more explicit

A character in a CTA 2063-A Serial Number "shall include any combination of digits and uppercase letters, except the letters O and I, but may include all digits". This would allow for a Base34 encoding of the binary HIT Suite ID and ORCHID hash. Although, ~~programatically~~programmatically, such a conversion is not hard, other technologies (e.g.~~,~~ credit card payment systems) that have used such odd base encoding have had performance challenges. Thus~~,~~ here a Base32 encoding will be used by also excluding the letters Z and S (too similar to the digits 2 and 5).

The low-order 68 bits (HIT Suite ID | ORCHID hash) of the HHIT SHALL be left-padded with 2 bits of ~~ZERO~~zeros. This ~~70~~ 70-bit number will be encoded into 14 characters using the digit/letters above. The manufacturer MUST use a Length Code of F (15). The first character after the Length Code MUST be 'Z', followed by the 14 characters of the encoded HIT Suite ID and ORCHID hash.

Commenté [BMI20]: We may add a note to remind the rationale

Commenté [BMI21]: We may add a note to remind the no rationale ;-))

Using the sample DET from Section 5 that is for HDA=20 under RAA=10 and having the ICAO CTA MFR Code of 8653, the ~~20~~ 20-character CTA 2063-A Serial Number would be:

8653FZ2T7B8RA85D19LX

A mapping service (e.g., ~~—~~DNS) MUST provide a trusted (e.g., via DNSSEC) conversion of the ~~4—~~4-character Manufacturer Code to high-order 60 bits (Prefix | HID) of the HHIT. Definition of this mapping service is currently out of scope of this document.

It should be noted that this encoding would only be used in the Basic ID Message. The HHIT DET will still be used in the Authentication Messages.

4.3. Remote ID DET as one class of Hierarchical HITs

UAS Remote ID DET may be one of a number of uses of HHITs. However, it is out of the scope of the document to elaborate on other uses of HHITs. As such these follow-on uses need to be considered in allocating the RAAs Section 3.3.1 or HHIT prefix assignments Section 9.

4.4. Hierarchy in ORCHID Generation

ORCHIDS, as defined in [RFC7343], do not cryptographically bind an IPv6 prefix nor the Orchid Generation Algorithm (OGA) ID (the HIT Suite ID) to the hash of the HI. The rational at the time of developing ORCHID was attacks against these fields are DoS attacks against protocols using ORCHIDs and thus up to those protocols to address the issue.

HHITs, as defined in Section 3.5, cryptographically bind all content in the ORCHID through the hashing function. A recipient of a DET that has the underlying HI can directly trust and act on all content in the HHIT. This provides a strong, self-attestation for using the hierarchy to find the DET Registry based on the HID.

4.5. DRIP Entity Tag (DET) Registry

DETs are registered to ~~Hierarchical HIT Domain Authorities (~~HDAs~~)~~. A registration process, [drip-registries], ensures DET global uniqueness (ID-4 in [drip-requirements]). It also provides the mechanism to create UAS ~~Public~~public/p~~P~~rivate data that are associated with the DET (REG-1 and REG-2 in [drip-requirements]).

The two levels of hierarchy within the DET 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.

4.6. Remote ID Authentication using DETs

The EdDSA25519 ~~Host Identity (~~HI~~)~~ ~~(~~[~~Section 3.4~~]~~)~~ underlying the DET can be used in an 84-byte ~~self-~~self-proof attestation (timestamp, HHIT, and signature of these) to provide proof of Remote ID ownership (GEN-1 in [drip-requirements]). In practice, the Wrapper and Manifest authentication formats in the ASTM Authentication Message (Msg Type 0x2) [drip-authentication] implicitly provide this self-attestation. A lookup service like DNS can provide the HI and registration proof (GEN-3 in [drip-requirements]).

~~Similarly~~Similarly, for Observers without Internet access, a 200-byte offline self-attestation could provide the same Remote ID ownership proof. This attestation would contain the HDA's signing of the UA's HHIT, itself signed by the UA's HI. Only a small cache that contains the HDA's HI/HHIT and HDA meta-data is needed by the Observer. However, such an object would just fit in the ASTM Authentication Message with no room for growth. In practice [drip-authentication] provides this offline self-attestation in two authentication messages: the HDA's certification of the UA's HHIT registration in a Link authentication message whose hash is sent in a Manifest authentication message.

Hashes of any previously sent ASTM messages can be placed in a Manifest authentication message (GEN-2 in [drip-requirements]). When a Location/Vector Message (Msg Type 0x1) hash along with the hash of the HDA's UA HHIT attestation are sent in a Manifest authentication message ~~AND~~ and the Observer can visually see a UA at the claimed location, the Observer has a very strong proof of the UA's Remote ID.

All this behavior and how to mix these authentication messages into the flow of UA operation messages are detailed in [drip-authentication].

5. DRIP Entity Tags (DETs) in DNS

There are two approaches for storing and retrieving ~~the~~ DETs using DNS~~. These are~~:

* As FQDNs in the "".aero" TLD.

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

A DET can be used to construct an FQDN that points to the USS that has the ~~Public~~public/~~Private~~ private information for the UA (REG-1 and REG-2 in [drip-requirements]). For example, the USS for the HHIT could be found via the following: Assume the RAA is 100 and the HDA is 50. The PTR record is constructed as follows:

100.50.det.uas.aero IN PTR foo.uss.aero.

The individual DETs ~~are~~ may be potentially too numerous (e.g., 60 - 600M) and dynamic (e.g., new DETs every minute for some HDAs) to ~~actually~~ ~~store~~store in a signed, DNS zone. The HDA SHOULD provide DNS service for its zone and provide the HHIT detail response. A secure connection (e.g.,~~-~~ DNS over TLS) to the authoritative zone may be a viable alternative to DNSSEC.

The DET reverse lookup can be a standard IPv6 reverse look up~~,~~ or it can leverage off the HHIT structure. If we ~~Assume~~ assume a prefix of 2001:30::/28, the RAA is 10, and the HDA is 20, ~~and~~ the DET is:

2001:30:a0:145:a3ad:1952:ad0:a69e

A DET reverse lookup could be to:

a69e.ad0.1952.a3ad.145.a0.30.2001.20.10.det.arpa.

or:

a3ad1952ad0a69e.5.20.10.30.2001.det.remoteid.aero.

A 'standard' ip6.arpa RR has the advantage of only one Registry service supported.

```
$ORIGIN 5.4.1.0.0.a.0.0.0.3.0.0.1.0.0.2.ip6.arpa.
e.9.6.a.0.d.a.0.2.5.9.1.d.a.3.a IN PTR
```

6. Other UTM uses of HHITs beyond DET

HHITs might be used within the UTM architecture beyond DET (and USS in UA ID registration and authentication). This includes a GCS HHIT ID. The GCS may use its HIIT if it is the source of Network Remote ID for securing the transport and for secure C2 transport (e.g., [drip-secure-nrid-c2]).

Observers may have their own 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 (this use is currently outside the scope of this document). Further, they can be used by FINDER observers, (e.g., [crowd-sourced-rid]).

Commenté [BMI22]: Add a note to explain the figures.

Commenté [BMI23]: Isn't this redundant with the text in Section 3.3.2?

Mis en forme : Surlignage

Commenté [BMI24]: Please check if this should be:

e.9.6.a.0.d.a.0.2.5.9.1.d.a.3.a.5.4.1.0.0.a.0.0.0.3.0.0.1.0.0.2.ip6.arpa.

Commenté [BMI25]: Please expand

7. DRIP Requirements addressed

This document in the previous sections provides the details to solutions for GEN 1 - 3, ID 1 - 5, and REG 1 - 2 as ~~described~~described in [drip-requirements].

8. DET Privacy

There is no expectation of privacy for DETs; it is not part of the Privacy Normative Requirements, Section 4.3.1, of [drip-requirements]. DETs are broadcast in the clear over the open air via Bluetooth and Wi-Fi. They will be collected and collated with other public information about the UAS. This will include DET registration information and location and times of operations for a DET. A DET can be for the life of a UA if there is no concern about DET/UA activity harvesting.

Further, the MAC address of the wireless interface used for Remote ID broadcasts are a target for UA operation aggregation that may not be mitigated through address randomization. For Bluetooth 4 Remote ID messaging, the MAC address is used by observers to link the Basic ID Message that contains the RID with other Remote ID messages, thus must be constant for a UA operation. This message linkage use of MAC addresses may not be needed with the Bluetooth 5 or Wi-Fi PHYs. These PHYs provide for a larger message payload and can use the Message Pack (Msg Type 0xF) and the Authentication Message to transmit the RID with other Remote ID messages. ~~However~~However, it is not ~~manditory~~mandatory to send the RID in a Message Pack or Authentication Message, so allowance for using the MAC address for UA message linking must be maintained. That is, the MAC address should be stable for at least a UA operation.

Finally, it is not adequate to simply change the DET and MAC for a UA per operation to defeat historically tracking a UA's activity.

Any changes to the UA MAC may have impacts to C2 setup and use. A constant GCS MAC may well defeat any privacy gains in UA MAC and RID changes. UA/GCS binding is complicated with changing MAC addresses; historically UAS design assumed these to be "forever" and made setup a one-time process. Additionally, if IP is used for C2, a changing MAC may mean a changing IP address to further impact the UAS bindings. Finally, an encryption wrapper's identifier (such as ESP [RFC4303] SPI) would need to change per operation to insure operation tracking separation.

Creating and maintaining UAS operational privacy is a multifaceted problem. Many communication pieces need to be considered to truly create a separation between UA operations. Simply changing the UAS RID only starts the changes that need to be implemented.

9. IANA Considerations

This document requests IANA to make the following changes to the IANA "Host Identity Protocol (HIP) Parameters" registry:

Host ID:
This document defines the new EdDSA Host ID with value TBD1 (suggested: 13) (~~see~~ Section 3.4.1) in the "HI Algorithm" subregistry of the "Host Identity Protocol (HIP) Parameters" registry.

EdDSA Curve Label:
This document specifies a new algorithm-specific subregistry named "EdDSA Curve Label". The values for this subregistry are defined in Section 3.4.1.

HIT Suite ID:
This document defines the new HIT Suite of EdDSA/cSHAKE with value TBD2 (suggested: 5) (see Section 3.4.2) in the "HIT Suite ID" subregistry of the "Host Identity Protocol (HIP) Parameters" registry.

HIT Suite ID eight-bit encoding:
This document defines the first eight-bit encoded HIT Suite IDs as defined in Section 5.2.10 of [RFC7401]. These are the new HDA domain HIT Suites with values TBD3 and TBD4 (suggested values: 0x0E and 0x0F) (~~see~~ Section 3.2.1). IANA is requested to expand the "HIT Suite ID" subregistry of the "Host Identity Protocol (HIP) Parameters" registry to show both the four-bit and eight-bit values as shown in Section 5.2.10 of [RFC7401] and add these new values that only have ~~eight~~ 8-bit representations.

9.1. New IPv6 prefix needed for HHITs

Because HHIT format is not compatible with [RFC7343], IANA is requested to ~~allocated~~allocate a new 28-bit prefix out of the IANA IPv6 Special Purpose Address Block, namely 2001:~~0000:~~:/23, as per [RFC6890] (suggested: 2001:30::/28).

Commenté [BMI26]: Please insist this is dedicated to UAD RID (DRIP)

10. Security Considerations

The 64-bit hash in HHITs presents a real risk of second pre-image cryptographic hash attack Section 10.2. There are no known (to the authors) studies of hash size to cryptographic hash attacks. A ~~PYTHON~~ Python script is available to randomly generate 1M HHITs that did not produce a hash collision which is a simpler attack than a first or second pre-image attack.

However, with today's computing power, producing 2^64 EdDSA keypairs and then generating the corresponding HHIT is economically feasible. Consider that a *single* bitcoin mining ASIC can do on the order of 2^46 sha256 hashes a second or about 2^62 hashes in a single day. The point being, 2^64 is not prohibitive, especially as this can be done in parallel.

Now it should be noted that the 2^64 attempts is for stealing a ~~*~~specific~~*~~ HHIT. Consider a scenario of a street photography company with 1,024 UAs (each with its own HHIT); you'd be happy stealing any one of them. Then rather than needing to satisfy a 64-bit condition on the cSHAKE128 output, you need only satisfy what is equivalent to a 54-bit condition (since you have 2^10 more opportunities for success).

Thus, although the ~~PROBABILITY~~ probability of a collision or pre-image attack is low in a collection of 1,024 HHITs out of a total population of 2^64, per Section 10.2, it is computationally and economically feasible. Thus, the HHIT registration and HHIT/HI registration validation is ~~STRONGLY~~ strongly recommended.

The DET Registry services effectively block attempts to "take over" or "hijack" a DET. It does not stop a rogue attempting to impersonate a known DET. This attack can be mitigated by the receiver of the DET using DNS to find the HI for the DET. As such, use of DNSSEC and DNS over TLS by the DET registries is recommended.

The ~~60-~~60-bit hash for DETs with ~~8-~~8-bit OGAs have a greater hash attack risk. As such its use should be restricted to testing and to small, well managed UAS/USS.

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 discussed in Section 4.6.

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., "det.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 [drip-registries], through its HHIT uniqueness enforcement, provides against forced or accidental HHIT hash collisions.

Cryptographically Generated Addresses (CGAs) provide an 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 ([drip-registries]) within the HDA provides a level of assured uniqueness unattainable without mirroring this approach.

The second aspect of assured uniqueness is the digital signing (attestation) process of the DET by the HI private key and the further signing (attestation) of the HI public key by the Registry's key. This completes the ownership process. The observer at this point does not know ~~WHAT~~ what owns the DET, 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.

10.1. DET Trust

The DET in the ASTM Basic ID 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 ID 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; that is, too open to a hash attack. Minimally, it takes 84 bytes, Section 4.6, to prove ownership of a DET with a full EdDSA signature. Thus, no attempt has been made to add DET trust directly within the very small Basic ID Message.

The ASTM Authentication Message (Msg Type 0x2) as shown in Section 4.6 can provide practical actual ownership proofs. These attestations 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 module 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 DET of the UA.

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

10.2. Collision risks with DETs

The ~~64~~ 64-bit hash size does have an increased risk of collisions over the ~~96~~ 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 B 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 DET and MUST reject a collision, forcing the UAS to generate a new HI and thus HHIT and reapplying to the DET registration process.

11. References

11.1. 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>.

[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>.

11.2. Informative References

[cfrg-comment] "A CFRG review of draft-ietf-drip-rid", September 2021, <https://mailarchive.ietf.org/arch/msg/cfrg/tAJJq60W6TlUv7_pde5cw5TDTCU/>.

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

[crowd-sourced-rid] Moskowitz, R., Card, S. W., Wiethuechter, A., Zhao, S., and H. Birkholz, "Crowd Sourced Remote ID", Work in Progress, Internet-Draft, draft-moskowitz-drip-crowd-sourced-rid-06, 26 May 2021, <https://datatracker.ietf.org/doc/html/draft-moskowitz-drip-crowd-sourced-rid-06>.

[CTA2063A] ANSI/CTA, "Small Unmanned Aerial Systems Serial Numbers", September 2019, <https://shop.cta.tech/products/small-unmanned-aerial-systems-serial-numbers>.

[drip-authentication] Wiethuechter, A., Card, S., and R. Moskowitz, "DRIP Authentication Formats for Broadcast RID", Work in Progress, Internet-Draft, draft-ietf-drip-auth-02, 21 October 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-drip-auth-02>.

[drip-registries] Wiethuechter, A., Card, S., and R. Moskowitz, "DRIP Registries", Work in Progress, Internet-Draft, draft-wiethuechter-drip-registries-01, 22 October 2021, <https://datatracker.ietf.org/doc/html/draft-wiethuechter-drip-registries-01>.

[drip-requirements] Card, S. W., Wiethuechter, A., Moskowitz, R., and A. Gurtov, "Drone Remote Identification Protocol (DRIP) Requirements", Work in Progress, Internet-Draft, draft-ietf-drip-reqs-18, 8 September 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-drip-reqs-18>.

[drip-secure-nrid-c2]
Moskowitz, R., Card, S. W., Wiethuechter, A., and A. Gurtov, "Secure UAS Network RID and C2 Transport", Work in Progress, Internet-Draft, draft-moskowitz-drip-secure-nrid-c2-04, 21 October 2021, <https://datatracker.ietf.org/doc/html/draft-moskowitz-drip-secure-nrid-c2-04>.

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

[FAA_RID] United States Federal Aviation Administration (FAA), "Remote Identification of Unmanned Aircraft", 2021, <https://www.govinfo.gov/content/pkg/FR-2021-01-15/pdf/2020-28948.pdf>.

[hhit-gen] "Python script to generate HHITs", September 2021, <https://github.com/ietf-wg-drip/draft-ietf-drip-rid/blob/master/hhit-gen.py>.

[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>.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <https://www.rfc-editor.org/info/rfc4303>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[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>.

[RFC7484] Blanchet, M., "Finding the Authoritative Registration Data (RDAP) Service", RFC 7484, DOI 10.17487/RFC7484, March 2015, <https://www.rfc-editor.org/info/rfc7484>.

[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>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

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. 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. Nicholas Gajcowski [cfrg-comment] provided a concise hash pre-image security assessment via the CFRG list.

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