Stream: Internet Engineering Task Force (IETF)
RFC: 9374
Updates: 7343, 7401
Category: Standards Track
Published: March 2023
ISSN: 2070-1721
Authors: R. Moskowitz, *HTT Consulting*; S. Card, *AX Enterprize, LLC*; A. Wiethuechter, *AX Enterprize, LLC*
A. Gurtov, *Linköping University*

RFC 9374
DRIP Entity Tag (DET) for Unmanned Aircraft System Remote ID (UAS RID)

Abstract

This document describes the use of Hierarchical Host Identity Tags (HHITs) as self-asserting IPv6 addresses, which makes them trustable identifiers for use in Unmanned Aircraft System Remote Identification (UAS RID) and tracking.

Within the context of RID, HHITs will be called DRIP Entity Tags (DETs). HHITs provide claims to the included explicit hierarchy that provides registry (via, for example, DNS, RDAP) discovery for third-party identifier endorsement.

This document updates RFCs 7343 and 7401.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9374.

Copyright Notice

Table of Contents

1. Introduction

Drone Remote ID Protocol (DRIP) Requirements [RFC9153] describe an Unmanned Aircraft System Remote ID (UAS ID) as unique (ID-4), non-spoofable (ID-5), and identify a registry where the ID is listed (ID-2); all within a 19-character identifier (ID-1).

This RFC is a foundational document of DRIP, as it 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 (see Section 3 of [DRIP-ARCH]). All other DRIP-related technologies will enable or use HHITs as multipurpose remote identifiers. HHITs add explicit hierarchy to the 128-bit HITs, enabling DNS HHIT queries (Host ID for authentication, e.g., [DRIP-AUTH]) and use with a Differentiated Access Control (e.g., Registration Data Access Protocol (RDAP) [RFC9224]) for 3rd-party identification endorsement (e.g., [DRIP-AUTH]).

The addition of hierarchy to HITs is an extension to [RFC7401] and requires an update to [RFC7343]. As this document also adds EdDSA (Section 3.4) for Host Identities (HIs), a number of Host Identity Protocol (HIP) parameters in [RFC7401] are updated, but these should not be needed in a DRIP implementation that does not use HIP.

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

HHITs provide self-claims of the HHIT registry. A HHIT can only be in a single registry within a registry system (e.g., DNS).

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

1.1. HHIT Statistical Uniqueness Different from UUID or X.509 Subject

HHITs are statistically unique through the cryptographic hash feature of second-preimage resistance. The cryptographically bound addition of the hierarchy and a HHIT registration process [DRIP-REG] provide complete, global HHIT uniqueness. If the HHITs cannot be looked up with services provided by the DRIP Identity Management Entity (DIME) identified via the embedded hierarchical information or its registration validated by registration endorsement messages [DRIP-AUTH], then the HHIT is either fraudulent or revoked/expired. In-depth discussion of these processes are out of scope for this document.

This contrasts with using general identifiers (e.g., Universally Unique IDentifiers (UUIDs) [RFC4122] or device serial numbers) as the subject in an X.509 [RFC5280] certificate. In either case, there can be no unique proof of ownership/registration.

For example, in a 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.

2. Terms and Definitions

2.1. Requirements Terminology

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**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.

The document includes a set of algorithms and recommends the ones that should be supported by implementations. The following term is used for that purpose: **RECOMMENDED**.

2.2. Notation

| 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 Section 2.2 of [RFC9153] and in Section 2 of [DRIP-ARCH]. The following terms are used in the document:

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

HDA (HHIT Domain Authority):
: The 14-bit field that identifies the HHIT Domain Authority under a Registered Assigning Authority (RAA). See Figure 1.

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 as defined in [RFC9063].

HID (Hierarchy ID):
: The 28-bit field providing the HIT Hierarchy ID. See Figure 1.

HIP (Host Identity Protocol):
: The origin of HI, HIT, and HHIT [RFC7401].

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. In particular, it refers to all the functions referenced from [NIST.FIPS.202] and [NIST.SP.800-185].

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

RAA (Registered Assigning Authority):
: The 14-bit field identifying the business or organization that manages a registry of HDAs. See Figure 1.

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

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 HHIT is a small but important enhancement over the flat Host Identity Tag (HIT) space, constructed as an Overlay Routable Cryptographic Hash IDentifier (ORCHID) [RFC7343]. By adding two levels of hierarchical administration control, the HHIT provides for device registration/ownership, thereby enhancing the trust framework for HITs.

The 128-bit HHITs represent the HI in only a 64-bit hash, rather than the 96 bits in HITs. 4 of these 32 freed up bits expand the Suite ID to 8 bits, and the other 28 bits are used to create a hierarchical administration organization for HIT domains. HHIT 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 (Figure 1):

- p = an IPv6 prefix (max 28 bit)
- 28-bit HID which provides the structure to organize HITs into administrative domains. HIDs are further divided into two fields:
  - 14-bit Registered Assigning Authority (RAA) (Section 3.3.1)
  - 14-bit HHIT Domain Authority (HDA) (Section 3.3.2)
- 8-bit HHIT Suite ID (HHSI)
- ORCHID hash (92 - prefix length, e.g., 64) See Section 3.5 for more details.

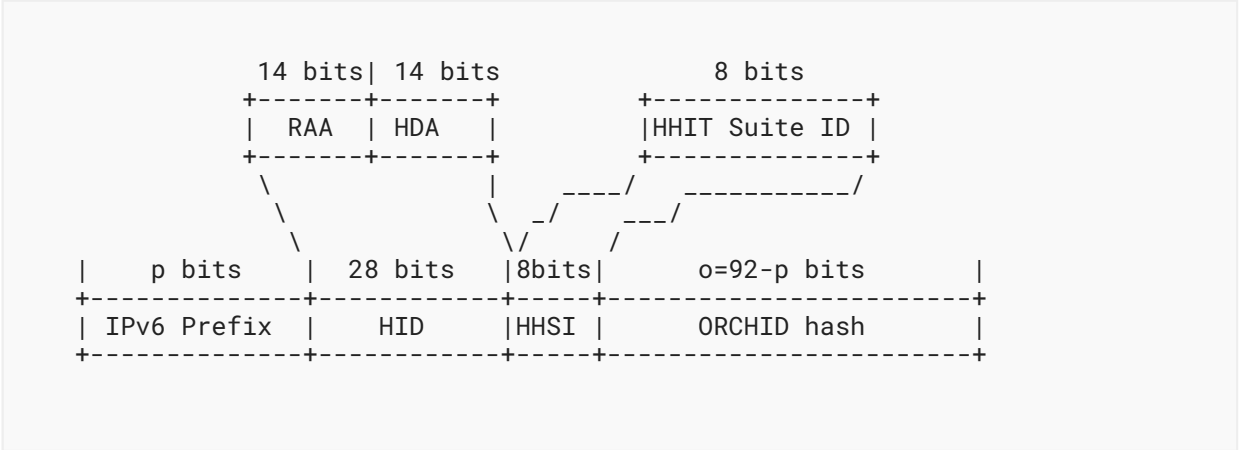

Figure 1: HHIT Format

The Context ID (generated with openssl rand) for the ORCHID hash is:

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

Context IDs are allocated out of the namespace introduced for Cryptographically Generated Addresses (CGA) Type Tags [RFC3972].

3.1. HHIT Prefix for RID Purposes

The IPv6 HHIT prefix **MUST** be distinct from that used in the flat-space HIT as allocated in [RFC7343]. Without this distinct prefix, the first 4 bits of the RAA would be interpreted as the HIT Suite ID per HIPv2 [RFC7401].

Initially, the IPv6 prefix listed in Table 1 is assigned for DET use. It has been registered in the "IANA IPv6 Special-Purpose Address Registry" [RFC6890].

HHIT Use	Bits	Value
DET	28	2001:30::/28

Table 1: Initial DET IPv6 Prefix

Other prefixes may be added in the future either for DET use or other applications of HHITs. For a prefix to be added to the registry in Section 8.2, its usage and HID allocation process have to be publicly available.

3.2. HHIT Suite IDs

The HHIT Suite IDs specify the HI and hash algorithms. These are a superset of the 4-bit and 8-bit HIT Suite IDs as defined in Section 5.2.10 of [RFC7401].

The HHIT values 1 - 15 map to the basic 4-bit HIT Suite IDs. HHIT values 17 - 31 map to the extended 8-bit HIT Suite IDs. HHIT values unique to HHIT will start with value 32.

As HHIT introduces a new Suite ID, EdDSA/cSHAKE128, and because this is of value to HIPv2, it will be allocated out of the 4-bit HIT space and result in an update to HIT Suite IDs. Future HHIT Suite IDs may be allocated similarly, or they may come out of the additional space made available by going to 8 bits.

The following HHIT Suite IDs are defined:

HHIT Suite	Value
RESERVED	0
RSA,DSA/SHA-256	1 [RFC7401]
ECDSA/SHA-384	2 [RFC7401]
ECDSA_LOW/SHA-1	3 [RFC7401]
EdDSA/cSHAKE128	5

Table 2: Initial HHIT Suite IDs

3.2.1. HDA Custom HIT Suite IDs

Support for 8-bit HHIT Suite IDs allows for HDA custom HIT Suite IDs (see Table 3).

HHIT Suite	Value
HDA Private Use 1	254
HDA Private Use 2	255

Table 3: HDA Custom HIT Suite IDs

These custom HIT Suite IDs, for example, may be used for large-scale experimentation with post-quantum computing hashes or similar domain-specific needs. Note that currently there is no support for domain-specific HI algorithms.

They should not be used to create a "de facto standardization". Section 8.2 states that additional Suite IDs can be made through IETF Review.

3.3. The Hierarchy ID (HID)

The HID provides the structure to organize HITs into administrative domains. HIDs are further divided into two fields:

- 14-bit Registered Assigning Authority (RAA)
- 14-bit HHIT Domain Authority (HDA)

The rationale for splitting the HID into two 14-bit domains is described in Appendix B.

The two levels of hierarchy allow for Civil Aviation Authorities (CAAs) to have it least one RAA for their National Air Space (NAS). Within its RAAs, the CAAs can delegate HDAs as needed. There may be other RAAs allowed to operate within a given NAS; this is a policy decision of each CAA.

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) or Japan Civil Aviation Bureau (JCAB) could be RAAs.

The RAA is a 14-bit field (16,384 RAAs). Management of this space is further described in [DRIP-REG]. An RAA **MUST** provide a set of services to allocate HDAs to organizations. It **SHOULD** have a public policy on what is necessary to obtain an HDA. The RAA need not maintain any HIP-related services. At minimum, it **MUST** maintain a DNS zone for the HDA zone delegation for discovering HIP RVS servers [RFC8004] for the HID. Zone delegation is covered in [DRIP-REG].

As DETs under administrative control may be used in many different domains (e.g., commercial, recreation, military), RAAs should be allocated in blocks (e.g., 16-19) with consideration of the likely size of a particular usage. Alternatively, different prefixes can be used to separate different domains of use of HHITs.

The RAA DNS zone within the UAS DNS tree may be a PTR for its RAA. It may be a zone in a HHIT-specific DNS zone. Assume that the RAA is decimal 100. The PTR record could be constructed as follows (where 20010030 is the DET prefix):

```
100.20010030.hhit.arpa.   IN PTR      raa.example.com.
```

Note that if the zone 20010030.hhit.arpa is ultimately used, a registrar will need to manage this for all HHIT applications. Thus, further thought will be needed in the actual DNS zone tree and registration process [DRIP-REG].

3.3.2. The HHIT Domain Authority (HDA)

An HDA may be an Internet Service Provider (ISP), UAS Service Supplier (USS), or any third party that takes on the business to provide UAS services management, HIP RVSs or other needed services such as those required for HHIT and/or HIP-enabled devices.

The HDA is a 14-bit field (16,384 HDAs per RAA) assigned by an RAA and is further described in [DRIP-REG]. An HDA must maintain public and private UAS registration information and should maintain a set of RVS servers for UAS clients that may use HIP. How this is done and scales to the potentially millions of customers are outside the scope of this document; they are covered in [DRIP-REG]. 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 a policy is out of scope for this document.

3.4. Edwards-Curve Digital Signature Algorithm for HHITs

The Edwards-Curve Digital Signature Algorithm (EdDSA) [RFC8032] is specified here for use as HIs per HIPv2 [RFC7401].

The intent in this document is to add EdDSA as a HI algorithm for DETs, but doing so impacts the HIP parameters used in a HIP exchange. Sections 3.4.1 through 3.4.2 describe the required updates to HIP parameters. Other than the HIP DNS RR (Resource Record) [RFC8005], these should not be needed in a DRIP implementation that does not use HIP.

See Section 3.2 for use of the HIT Suite in the context of DRIP.

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 5.2.9 of [RFC7401]. Table 4 adds a new HI Algorithm.

Algorithm profile	Value	Reference
EdDSA	13	[RFC8032]

Table 4: New EdDSA Host ID

3.4.1.1. HIP Parameter support for EdDSA

The addition of EdDSA as a HI algorithm requires a subfield in the HIP HOST_ID parameter (Section 5.2.9 of [RFC7401]) as was done for ECDSA when used in a HIP exchange.

For HIP hosts that implement EdDSA as the algorithm, the following EdDSA curves are represented by the fields in Figure 2.

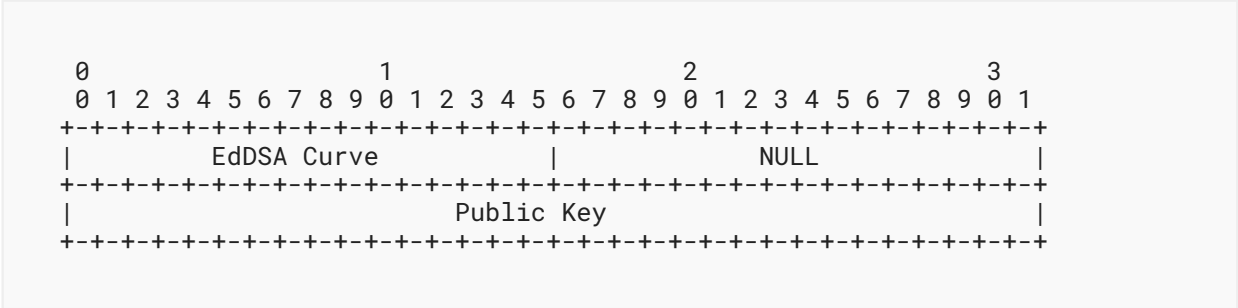

Figure 2: EdDSA Curves Fields

EdDSA Curve: Curve label

Public Key: Represented in Octet-string format [RFC8032]

For hosts that implement EdDSA as a HIP algorithm, the following EdDSA curves are defined. Recommended curves are tagged accordingly:

Algorithm	Curve	Values
EdDSA	RESERVED	0
EdDSA	EdDSA25519	1 [RFC8032] (**RECOMMENDED**)
EdDSA	EdDSA25519ph	2 [RFC8032]
EdDSA	EdDSA448	3 [RFC8032] (**RECOMMENDED**)
EdDSA	EdDSA448ph	4 [RFC8032]

Table 5: EdDSA Curves

3.4.1.2. HIP DNS RR support for EdDSA

The HIP DNS RR is defined in [RFC8005]. It uses the values defined for the 'Algorithm Type' of the IPSECKEY RR [RFC4025] for its PK Algorithm field.

The 'Algorithm Type' value and EdDSA HI encoding are assigned per [RFC9373].

3.4.2. HIT_SUITE_LIST

The HIT_SUITE_LIST parameter contains a list of the HIT suite IDs that the HIP Responder supports. The HIT_SUITE_LIST allows the HIP Initiator to 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:

HIT Suite	Value
EdDSA/cSHAKE128	5

Table 6: HIT Suite ID

Table 7 provides more detail on the above HIT Suite combination.

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 7: HIT Suites

3.5. ORCHIDs for HHITs

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

- the inclusion of an optional "Info" field between the Prefix and ORCHID Generation Algorithm (OGA) ID.
- an increase in flexibility on the length of each component in the ORCHID construction, provided the resulting ORCHID is 128 bits.
- the use of cSHAKE [NIST.SP.800-185] for the hashing function.

The cSHAKE XOF hash function based on Keccak [Keccak] 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 HHIT) in the ORCHID generation. This should be viewed as an update to ORCHIDv2 [RFC7343], as it can produce ORCHIDv2 output.

The following subsections define the new general ORCHID construct with the specific application for HHITs. Thus items like the hash size are only discussed in terms of how they impact the HHIT's 64-bit hash. Other hash sizes should be discussed for other specific uses of this new ORCHID construct.

3.5.1. Adding Additional Information to the ORCHID

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

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

where:

Prefix
: A constant 28-bit-long bitstring value (IPv6 prefix)

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.

The new ORCHID function is as follows:

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

where:

Prefix (p)
: An IPv6 prefix of length p (max 28 bits long).

Info (n)
: n bits of information that define a use of the ORCHID. 'n' can be zero, which means 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 [IANA-HIP]. When used for HHIT generation, this is the HHIT Suite ID [HHSI].

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

Sizeof(p + n + o + m) = 128 bits

The ORCHID length **MUST** be 128 bits. For HHITs with a 28-bit IPv6 prefix, there are 100 bits remaining to be divided in any manner between the additional information ("Info"), OGA ID, and the hash output. Consideration must be given to the size of the hash portion, taking into account

risks like pre-image attacks. 64 bits, as used here for HHITs, may be as small as is acceptable. The size of 'n', for the HID, is then determined as what is left; in the case of the 8-bit OGA used for HHIT, this is 28 bits.

3.5.2. ORCHID Encoding

This update 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 ("Info"), and the OGA ID (HIT Suite ID). Secondly, the length of the resulting hash is set by the sum of the length of the ORCHID header fields. For example, a 28-bit prefix with 28 bits for the HID and 8 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 cSHAKE function call is:

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

Input      :=  Prefix | Additional Information | OGA ID | HOST_ID
L          :=  Length in bits of the 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 a /28 prefix and Additional Information of zero length):

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

Input      :=  Prefix | Additional Information | OGA ID | HOST_ID
L          :=  Length in bits of the 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.
```

The middle L-bits are those bits from the source number where either there is an equal number of bits before and after these bits, or there is one more bit prior (when the difference between hash size and L is odd).

HHITs use the Context ID defined in Section 3.

3.5.2.1. Encoding ORCHIDs for HIPv2

This section discusses how to provide backwards compatibility for ORCHIDv2 [RFC7343] as used in HIPv2 [RFC7401].

For HIPv2, the Prefix is 2001:20::/28 (Section 6 of [RFC7343]). 'Info' is zero-length (i.e., not included), and OGA ID is 4-bit. Thus, the HI Hash is 96 bits in length. 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
```

3.5.3. ORCHID Decoding

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

For HHITs, the decoding is determined by the presence of the HHIT Prefix as specified in Section 8.2.

3.5.4. Decoding ORCHIDs for HIPv2

This section is included to provide backwards compatibility for ORCHIDv2 [RFC7343] as used for HIPv2 [RFC7401].

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

4. HHITs as DRIP Entity Tags

HHITs for UAS ID (called, DETs) use the new EdDSA/SHAKE128 HIT suite defined in Section 3.4 (GEN-2 in [RFC9153]). This hierarchy, cryptographically bound within the HHIT, provides the information for finding the UA's HHIT registry (ID-3 in [RFC9153]).

The ASTM Standard Specification for Remote ID and Tracking [F3411-22a] adds support for DETs. This is only available via the new UAS ID type 4, "Specific Session ID (SSI)".

This new SSI uses the first byte of the 20-byte UAS ID for the SSI Type, thus restricting the UAS ID of this type to a maximum of 19 bytes. The SSI Types initially assigned are:

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

SSI 2: 3GPP - IEEE 1609.2-2016 HashedID8

4.1. Nontransferablity of DETs

A HI and its DET **SHOULD NOT** be transferable between UAs 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, 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 HHIT within the CTA 2063-A format is not simple. The CTA 2063-A format is defined as follows:

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

where:

MFR Code
: 4 character code assigned by ICAO (International Civil Aviation Organization, a UN Agency).

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

MFR SN
: US-ASCII 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 HHIT Suite ID and ORCHID hash will take the full 15 characters (as described below) of the MFR SN field.

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 HHIT Suite ID and ORCHID hash in 15 characters. Although, 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 (because they are too similar to the digits 2 and 5, respectively). See Appendix C for the encoding scheme.

The low-order 72 bits (HHIT Suite ID | ORCHID hash) of the HHIT **SHALL** be left-padded with 3 bits of zeros. This 75-bit number will be encoded into the 15-character MFR SN field using the digit/letters as described above. The manufacturer **MUST** use a Length Code of F (15).

Note: The manufacturer **MAY** use the same Manufacturer Code with a Length Code of 1 - E (1 - 14) for other types of serial numbers.

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-character CTA 2063-A Serial Number would be:

```
8653F02T7B8RA85D19LX
```

A mapping service (e.g., DNS) **MUST** provide a trusted (e.g., via DNSSEC [RFC4034]) conversion of the 4-character Manufacturer Code to high-order 58 bits (Prefix | HID) of the HHIT. That is, given a Manufacturer Code, a returned Prefix|HID value is reliable. Definition of this mapping service is out of scope of this document.

It should be noted that this encoding would only be used in the Basic ID Message (Section 2.2 of [RFC9153]). The DET is used in the Authentication Messages (i.e., the messages that provide framing for authentication data only).

4.3. Remote ID DET as one Class of HHITs

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 8).

4.4. Hierarchy in ORCHID Generation

ORCHIDS, as defined in [RFC7343], do not cryptographically bind an IPv6 prefix or the OGA ID (the HIT Suite ID) to the hash of the HI. At the time ORCHID was being developed, the rationale was attacks against these fields are Denial-of-Service (DoS) attacks against protocols using ORCHIDs and thus it was 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-claim for using the hierarchy to find the DET Registry based on the HID (Section 4.5).

4.5. DRIP Entity Tag (DET) Registry

DETs are registered to HDAs. The registration process defined in [DRIP-REG] ensures DET global uniqueness (ID-4 in Section 4.2.1 of [RFC9153]). It also allows the mechanism to create UAS public/private data that are associated with the DET (REG-1 and REG-2 in Section 4.4.1 of [RFC9153]).

4.6. Remote ID Authentication Using DETs

The EdDSA25519 HI (Section 3.4) underlying the DET can be used in an 88-byte self-proof evidence (timestamps, HHIT, and signature of these) to provide proof to Observers of Remote ID ownership (GEN-1 in Section 4.1.1 of [RFC9153]). In practice, the Wrapper and Manifest authentication formats (Sections 6.3.3 and 6.3.4 of [DRIP-AUTH]) implicitly provide this self-proof evidence. A lookup service like DNS can provide the HI and registration proof (GEN-3 in [RFC9153]).

Similarly, for Observers without Internet access, a 200-byte offline self-endorsement (Section 3.1.2 of [DRIP-AUTH]) could provide the same Remote ID ownership proof. This endorsement would contain the HDA's signing of the UA's HHIT, itself signed by the UA's HI. Only a small cache (also Section 3.1.2 of [DRIP-AUTH]) 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 (Section 2.2 of [RFC9153]) with no room for growth. In practice, [DRIP-AUTH] provides this offline self-endorsement in two authentication messages: the HDA's endorsement 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 [RFC9153]). When a Location/Vector Message (i.e., a message that provides UA location, altitude, heading, speed, and status) hash along with the hash of the HDA's UA HHIT endorsement are sent in a Manifest authentication message and the Observer can visually see a UA at the claimed location, the Observer has very strong proof of the UA's Remote ID.

This behavior and how to mix these authentication messages into the flow of UA operation messages are detailed in [DRIP-AUTH].

5. DRIP Entity Tags (DETs) in DNS

There are two approaches for storing and retrieving DETs using DNS. The following are examples of how this may be done. This serves as guidance to the actual deployment of DETs in DNS. However, this document does not provide a recommendation about which approach to use. Further DNS-related considerations are covered in [DRIP-REG].

- As FQDNs, for example, "20010030.hhit.arpa.".
- 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/private information for the UA (REG-1 and REG-2 in Section 4.4.1 of [RFC9153]). For example, the USS for the HHIT could be found via the following: assume the RAA is decimal 100 and the HDA is decimal 50. The PTR record is constructed as follows:

```
100.50.20010030.hhit.arpa.   IN PTR      foo.uss.example.org.
```

The HDA **SHOULD** provide DNS service for its zone and provide the HHIT detail response.

The DET reverse lookup can be a standard IPv6 reverse look up, or it can leverage off the HHIT structure. Using the allocated prefix for HHITs 2001:30::/28 (see Section 3.1), the RAA is decimal 10 and the HDA is decimal 20, the DET is:

```
2001:30:280:1405:a3ad:1952:ad0:a69e
```

See Appendix B.1 for how the upper 64 bits, above, are constructed. A DET reverse lookup could be:

```
a69e.0ad0.1952.a3ad.1405.0280.20.10.20010030.hhit.arpa.
```

or:

```
a3ad19520ad0a69e.5.20.10.20010030.hhit.arpa.
```

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

```
$ORIGIN  5.0.4.1.0.8.2.0.0.3.0.0.1.0.0.2.ip6.arpa.
e.9.6.a.0.d.a.0.2.5.9.1.d.a.3.a    IN   PTR
a3ad1952ad0a69e.20.10.20010030.hhit.arpa.
```

This DNS entry for the DET can also provide a revocation service. For example, instead of returning the HI RR it may return some record showing that the HI (and thus DET) has been revoked. Guidance on revocation service will be provided in [DRIP-REG].

6. Other UAS Traffic Management (UTM) Uses of HHITs Beyond DET

HHITs will be used within the UTM architecture beyond DET (and USS in UA ID registration and authentication), for example, as a Ground Control Station (GCS) HHIT ID. Some GCS will use its HHIT for securing its Network Remote ID (to USS HHIT) and Command and Control (C2, Section 2.2.2 of [RFC9153]) transports.

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. Details about such issues are out of the scope of this document.

7. Summary of Addressed DRIP Requirements

This document provides the details to solutions for GEN 1 - 3, ID 1 - 5, and REG 1 - 2 requirements that are described in [RFC9153].

8. IANA Considerations

8.1. New Well-Known IPv6 Prefix for DETs

Since the DET format is not compatible with [RFC7343], IANA has allocated the following prefix per this template for the "IANA IPv6 Special-Purpose Address Registry" [IPv6-SPECIAL].

Address Block:
2001:30::/28

Name:
Drone Remote ID Protocol Entity Tags (DETs) Prefix

Reference
This document

Allocation Date:
2022-12

Termination Date:
N/A

Source:
True

Destination:
True

Forwardable:
True

Globally Reachable:
True

Reserved-by-Protocol:
False

8.2. New IANA DRIP Registry

IANA has created the "Drone Remote ID Protocol" registry. The following two subregistries have been created within the "Drone Remote ID Protocol" group.

8.2.1. HHIT Prefixes

Initially, for DET use, one 28-bit prefix has been assigned out of the IANA IPv6 Special Purpose Address Block, namely 2001::/23, as per [RFC6890]. Future additions to this subregistry are to be made through Expert Review (Section 4.5 of [RFC8126]). Entries with network-specific prefixes may be present in the registry.

HHIT Use	Bits	Value	Reference
DET	28	2001:30::/28	RFC 9374

Table 8: Registered DET IPv6 Prefix

Criteria that should be applied by the designated experts includes determining whether the proposed registration duplicates existing functionality and whether the registration description is clear and fits the purpose of this registry.

Registration requests **MUST** be sent to drip-reg-review@ietf.org and be evaluated within a three-week review period on the advice of one or more designated experts. Within that review period, the designated experts will either approve or deny the registration request, and communicate their decision to the review list and IANA. Denials should include an explanation and, if applicable, suggestions to successfully register the prefix.

Registration requests that are undetermined for a period longer than 28 days can be brought to the IESG's attention for resolution.

8.2.2. HHIT Suite IDs

This 8-bit value subregistry is a superset of the 4/8-bit "HIT Suite ID" subregistry of the "Host Identity Protocol (HIP) Parameters" registry [IANA-HIP]. Future additions to this subregistry are to be made through IETF Review (Section 4.8 of [RFC8126]). The following HHIT Suite IDs are defined.

HHIT Suite	Value	Reference
RESERVED	0	RFC 9374
RSA,DSA/SHA-256	1	[RFC7401]
ECDSA/SHA-384	2	[RFC7401]
ECDSA_LOW/SHA-1	3	[RFC7401]
EdDSA/cSHAKE128	5	RFC 9374
HDA Private Use 1	254	RFC 9374
HDA Private Use 2	255	RFC 9374

Table 9: Registered HHIT Suite IDs

The HHIT Suite ID values 1 - 31 are reserved for IDs that **MUST** be replicated as HIT Suite IDs (Section 8.4) as is 5 here. Higher values (32 - 255) are for those Suite IDs that need not or cannot be accommodated as a HIT Suite ID.

8.3. IANA CGA Registry Update

This document has been added as a reference for the "CGA Extension Type Tags" registry [IANA-CGA]. IANA has the following Context ID in this registry:

Context ID:
: The Context ID (Section 3) shares the namespace introduced for CGA Type Tags. The following Context ID is defined per the rules in Section 8 of [RFC3972]:

CGA Type Tag	Reference
0x00B5 A69C 795D F5D5 F008 7F56 843F 2C40	RFC 9374

Table 10: CGA Extension Type Tags

8.4. IANA HIP Registry Updates

IANA has updated the "Host Identity Protocol (HIP) Parameters" registry [IANA-HIP] as described below.

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

Algorithm Profile	Value	Reference
EdDSA	13	[RFC8032]

Table 11: Registered HI Algorithm

EdDSA Curve Label:
: This document specifies a new algorithm-specific subregistry named "EdDSA Curve Label". The values for this subregistry are defined in Section 3.4.1.1. Future additions to this subregistry are to be made through IETF Review (Section 4.8 of [RFC8126]).

Algorithm	Curve	Value	Reference
EdDSA	RESERVED	0	RFC 9374
EdDSA	EdDSA25519	1	[RFC8032]
EdDSA	EdDSA25519ph	2	[RFC8032]
EdDSA	EdDSA448	3	[RFC8032]
EdDSA	EdDSA448ph	4	[RFC8032]
		5-65535	Unassigned

Table 12: Registered EdDSA Curve Labels

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

Suite ID	Value	Reference
EdDSA/cSHAKE128	5	RFC 9374

Table 13: Registered HIT Suite of EdDSA/cSHAKE

The HIT Suite ID 4-bit values 1 - 15 and 8-bit values 0x00 - 0x0F **MUST** be replicated as HHIT Suite IDs (Section 8.2) as is 5 here.

9. Security Considerations

The 64-bit hash in HHITs presents a real risk of second pre-image cryptographic hash attack (see Section 9.5). There are no known (to the authors) studies of hash size impact on cryptographic hash attacks.

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 per 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.

Note 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); an attacker may well be satisfied stealing any one of them. Then, rather than needing to satisfy a 64-bit condition on the cSHAKE128 output, an attacker only needs to satisfy what is equivalent to a 54-bit condition (since there are 2^{10} more opportunities for success).

Thus, although the 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 9.5), it is computationally and economically feasible. Therefore, the HHIT registration is a **MUST** and HHIT/HI registration validation **SHOULD** be performed by Observers either through registry lookups or via broadcasted registration proofs (Section 3.1.2 of [DRIP-AUTH]).

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 messages containing DETs using DNS to find the HI for the DET. As such, use of DNSSEC by the DET registries is recommended to provide trust in HI retrieval.

Another mitigation of HHIT hijacking is when the HI owner (UA) supplies an object containing the HHIT that is signed by the HI private key of the HDA as detailed in [DRIP-AUTH].

The two risks with HHITs are the use of an invalid HID and forced HIT collisions. The use of a DNS zone (e.g., "det.arpa.") is 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-REG], 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 (and more importantly, the implied second-preimage resistance) makes attacks statistically challenging. A registration process [DRIP-REG] within the HDA provides a level of assured uniqueness unattainable without mirroring this approach.

The second aspect of assured uniqueness is the digital signing (evidence) process of the DET by the HI private key and the further signing (evidence) 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 DET but is assured, other than the risk of theft of the HI private key, that this UAS ID is owned by something and it is properly registered.

9.1. Post-Quantum Computing Is Out of Scope

As stated in Section 8.1 of [DRIP-ARCH], there has been no effort to address post-quantum computing cryptography. UAs and Broadcast Remote ID communications are so constrained that current post-quantum computing cryptography is not applicable. In addition, because a UA may use a unique DET for each operation, the attack window could be limited to the duration of the operation.

HHITs contain the ID for the cryptographic suite used in its creation, a future algorithm that is safe for post-quantum computing that fits the Remote ID constraints may readily be added.

9.2. DET Trust in ASTM Messaging

The DET in the ASTM Basic ID Message (Msg Type 0x0, the actual Remote ID message) does not provide any assertion of trust. Truncating 4 bytes from a HI signing of the HHIT (the UA ID field is 20 bytes and a HHIT is 16) within this Basic ID Message is the best that can be done. This is not trustable, as it is too open to a hash attack. Minimally, it takes 88 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 actual ownership proofs in a practical manner. The endorsements and evidence include timestamps to defend against replay attacks, but they do not prove which UA sent the message. The messages 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, for example, when the Authentication Message includes proof of the ASTM Location/Vector Message (Msg Type 0x1) and a) the observer can see the UA or b) the location information is validated by ground multilateration. Only then does an observer gain full trust in the DET of the UA.

DETs obtained via the Network RID path provide a different approach to trust. Here the UAS **SHOULD** be securely communicating to the USS, thus asserting DET trust.

9.3. DET Revocation

The DNS entry for the DET can also provide a revocation service. For example, instead of returning the HI RR, it may return some record showing that the HI (and thus DET) has been revoked. Guidance on revocation service will be provided in [DRIP-REG].

9.4. Privacy Considerations

There is no expectation of privacy for DETs; it is not part of the normative privacy requirements listed in Section 4.3.1 of [RFC9153]. 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 Media Access Control (MAC) address of the wireless interface used for Remote ID broadcasts are a target for UA operation aggregation that may not be mitigated through MAC 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 it must be constant for a UA operation. This use of MAC addresses to link messages 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, sending the RID in a Message Pack or Authentication Message is not mandatory, so using the MAC address for UA message linking must be allowed. 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 tracking the history of the 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 if the UA MAC address can change; 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 ensure 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. Changing the DET is only the start of the changes that need to be implemented.

These privacy realities may present challenges for the European Union (EU) U-space (Appendix A) program.

9.5. Collision Risks with DETs

The 64-bit hash size here for DETs does have an increased risk of collisions over the 96-bit hash size used for the ORCHID [RFC7343] construct. 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 D for the collision probability formula.

However, this risk of collision is within a single "Additional Information" value, i.e., an 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 (Section 6 of [DRIP-REG]).

Thus an adversary trying to generate a collision and 'steal' the DET would run afoul of this registration process and associated validation process mentioned in Section 1.1.

10. References

10.1. Normative References

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

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

[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>.

[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>.

[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>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[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>.

[RFC9373] Moskowitz, R., Kivinen, T., and M. Richardson, "EdDSA Value for IPSECKEY", RFC 9373, DOI 10.17487/RFC9373, March 2023, <https://www.rfc-editor.org/info/rfc9373>.

10.2. Informative References

[CFRG-COMMENT] Gajcowski, N., "Please review draft-ietf-drip-rid", message to the CFRG mailing list, 23 September 2021, <https://mailarchive.ietf.org/arch/msg/cfrg/tAJJq60W6TlUv7_pde5cw5TDTCU/>.

[CORUS] CORUS, "SESAR Concept of Operations for U-space", 9 September 2019, <https://www.sesarju.eu/node/3411>.

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

[DRIP-ARCH] Card, S. W., Wiethuechter, A., Moskowitz, R., Zhao, S., and A. Gurtov, "Drone Remote Identification Protocol (DRIP) Architecture", Work in Progress, Internet-Draft, draft-ietf-drip-arch-31, 6 March 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-drip-arch-31>.

[DRIP-AUTH] Wiethuechter, A., Card, S. W., and R. Moskowitz, "DRIP Entity Tag Authentication Formats & Protocols for Broadcast Remote ID", Work in Progress, Internet-Draft, draft-ietf-drip-auth-29, 15 February 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-drip-auth-29>.

[DRIP-REG] Wiethuechter, A. and J. Reid, "DRIP Entity Tag (DET) Identity Management Architecture", Work in Progress, Internet-Draft, draft-ietf-drip-registries-07, 5 December 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-drip-registries-07>.

[F3411-22a] ASTM International, "Standard Specification for Remote ID and Tracking - F3411-22a", July 2022, <https://www.astm.org/f3411-22a.html>.

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

[HHSI] IANA, "Hierarchical HIT (HHIT) Suite IDs", <https://www.iana.org/assignments/drip>.

[IANA-CGA] IANA, "Cryptographically Generated Addresses (CGA) Message Type Name Space", <https://www.iana.org/assignments/cga-message-types>.

[IANA-HIP] IANA, "Host Identity Protocol (HIP) Parameters", <https://www.iana.org/assignments/hip-parameters>.

[IPv6-SPECIAL] IANA, "IANA IPv6 Special-Purpose Address Registry", <https://www.iana.org/assignments/iana-ipv6-special-registry/>.

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

[RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, DOI 10.17487/RFC3972, March 2005, <https://www.rfc-editor.org/info/rfc3972>.

[RFC4025] Richardson, M., "A Method for Storing IPsec Keying Material in DNS", RFC 4025, DOI 10.17487/RFC4025, March 2005, <https://www.rfc-editor.org/info/rfc4025>.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <https://www.rfc-editor.org/info/rfc4034>.

[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>.

[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>.

[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>.

[RFC9063] Moskowitz, R., Ed. and M. Komu, "Host Identity Protocol Architecture", RFC 9063, DOI 10.17487/RFC9063, July 2021, <https://www.rfc-editor.org/info/rfc9063>.

[RFC9153] Card, S., Ed., Wiethuechter, A., Moskowitz, R., and A. Gurtov, "Drone Remote Identification Protocol (DRIP) Requirements and Terminology", RFC 9153, DOI 10.17487/RFC9153, February 2022, <https://www.rfc-editor.org/info/rfc9153>.

[RFC9224] Blanchet, M., "Finding the Authoritative Registration Data Access Protocol (RDAP) Service", STD 95, RFC 9224, DOI 10.17487/RFC9224, March 2022, <https://www.rfc-editor.org/info/rfc9224>.

Appendix A. EU U-Space RID Privacy Considerations

The EU is defining a future of airspace management known as U-space within the Single European Sky ATM Research (SESAR) undertaking. The Concept of Operation for EuRopean UTM Systems (CORUS) project proposed low-level Concept of Operations [CORUS] for UAS in the EU. It introduces strong requirements for UAS privacy based on European General Data Protection Regulation (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.

This is potentially a contrary expectation as that presented in Section 9.4. U-space will have to deal with this reality within the GDPR regulations. Still, DETs as defined here present a large step in the right direction for agnostic IDs.

The project lists "E-Identification" and "E-Registrations" services as to be developed. These services can use DETs and follow the privacy considerations outlined in this document for DETs.

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 DET (HHIT) provides 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 14/14 HID split

The following explains the logic for dividing the 28 bits of the HID into two 14-bit components.

At this writing, the International Civil Aviation Organization (ICAO) has 193 member "States", and each may want to control RID assignment within its National Air Space (NAS). Some members may want separate RAAs to use for Civil, general Government, and Military use. They may also want allowances for competing Civil RAA operations. It is reasonable to plan for eight RAAs per ICAO member (plus regional aviation organizations like in the EU). Thus, as a start, a space of 4,096 RAAs is advised.

There will be requests by commercial entities for their own RAA allotments. Examples could include international organizations that will be using UAS and international delivery service associations. These may be smaller than the RAA space needed by ICAO member States and could be met with a 2,048 space allotment; however, as will be seen, these might as well be 4,096 as well.

This may well cover currently understood RAA entities. In the future, there will be new applications, branching off into new areas, so yet another space allocation should be set aside. If this is equal to all that has been reserved, we should allow for 16,384 (2^{14}) RAAs.

The HDA allocation follows a different logic from that of RAAs. Per Appendix D, an HDA should be able to easily assign 63M RIDs and even manage 663M with a "first come, first assigned" registration process. For most HDAs, this is more than enough, and a single HDA assignment within their RAA will suffice. Most RAAs will only delegate to a couple of HDAs for their operational needs. But there are major exceptions that point to some RAAs needing large numbers of HDA assignments.

Delivery service operators like Amazon (est. 30K delivery vans) and UPS (est. 500K delivery vans) may choose, for anti-tracking reasons, to use unique RIDs per day or even per operation. 30K delivery UAs could need between 11M and 44M RIDs. Anti-tracking would be hard to provide if the HID were the same for a delivery service fleet, so such a company may turn to an HDA that provides this service to multiple companies so that who's UA is who's is not evident in the HID. A USS providing this service could well use multiple HDA assignments per year, depending on strategy.

Perhaps a single RAA providing HDAs for delivery service (or a similar purpose) UAS could 'get by' with a 2048 HDA space (11 bits). So the HDA space could well be served with only 12 bits allocated out of the 28-bit HID space. However, as this is speculation and deployment experience will take years, a 14-bit HDA space has been selected.

There may also be 'small' ICAO member States that opt for a single RAA and allocate their HDAs for all UAs that are permitted in their NAS. The HDA space is large enough that a portion may be used for government needs as stated above and small commercial needs. Alternatively, the State may use a separate, consecutive RAA for commercial users. Thus it would be 'easy' to recognize State-approved UA by HID high-order bits.

B.1. DET Encoding Example

The upper 64 bits of DET appear to be oddly constructed from nibbled fields, when typically seen in 8-bit representations. The following works out the construction of the example in Section 5.

In that example, the prefix is 2001:30::/28, the RAA is decimal 10, and the HDA is decimal 20. Below is the RAA and HDA in 14-bit format:

```
RAA 10 = 00000000001010
HDA 20 = 00000000010100
```

The leftmost 4 bits of the RAA, all zeros, combine with the prefix to form 2001:0030:, which leaves the remaining RAA and HDA to combine to:

```
0000|0010|1000|0000|0001|0100|
```

Which when combined with the OGA of x05 is 0280:1405, thus the whole upper 64 bits are 2001:0030:0280:1405.

Appendix C. Base32 Alphabet

The alphabet used in CTA 2063-A Serial Number does not map to any published Base32 encoding scheme. Therefore, the following Base32 Alphabet is used.

Each 5-bit group is used as an index into an array of 32 printable characters. The character referenced by the index is placed in the output string. These characters, identified below, are selected from US-ASCII digits and uppercase letters.

Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
0	0	8	8	16	G	24	Q
1	1	9	9	17	H	25	R
2	2	10	A	18	J	26	T
3	3	11	B	19	K	27	U
4	4	12	C	20	L	28	V
5	5	13	D	21	M	29	W
6	6	14	E	22	N	30	X
7	7	15	F	23	P	31	Y

Table 14: The Base 32 Alphabet

Appendix D. 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

Table 15: Approximate Population Size With Collision Risk

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 supporting NIST 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.

Many thanks to Michael Richardson and Brian Haberman for the iotdir review, Magnus Nystrom for the secdir review, Elwyn Davies for the genart review, and the DRIP co-chair and document shepherd, Mohamed Boucadair for his extensive comments and help on document clarity. And finally, many thanks to the Area Directors: Roman Danyliw, Erik Kline, Murray Kucherawy, Warren Kumari, John Scudder, Paul Wouters, and Sarker Zaheduzzaman, for the IESG review.

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