          An IPv6 Prefix for Overlay Routable Cryptographic Hash Identifiers
                          Version 2 (ORCHIDv2)
                      draft-ietf-hip-rfc4843-bis-03

Abstract

   This document specifies an updated Overlay Routable Cryptographich
   Hash Identifiers format that obsoletes the earlier format defined in
   [RFC4843]. These identifiers are intended to be used as endpoint
   identifiers at applications and Application Programming Interfaces
   (API) and not as identifiers for network location at the IP layer,
   i.e., locators. They are designed to appear as application layer
   entities and at the existing IPv6 APIs, but they should not appear in
   actual IPv6 headers. To make them more like vanilla IPv6 addresses,
   they are expected to be routable at an overlay level. Consequently,
   while they are considered non-routable addresses from the IPv6 layer
   point-of-view, all existing IPv6 applications are expected to be able
   to use them in a manner compatible with current IPv6 addresses.

   The Overlay Routable Cryptographic Hash Identifiers originally
   defined in [RFC4843] lacked a mechanism for cryptographic algorithm
   agility. The updated ORCHID format specified in this document
   removes this limitation by encoding in the identifier itself an index
   to the suite of cryptographic algorithms in use.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

   This document introduces Overlay Routable Cryptographic Hash
   Identifiers (ORCHID), a new class of IP address-like identifiers.
   These identifiers are intended to be globally unique in a statistical
   sense (see Section 4), non-routable at the IP layer, and routable at
   some overlay layer. The identifiers are securely bound, via a secure
   hash function, to the concatenation of an input bitstring and a
   context tag. Typically, but not necessarily, the input bitstring
   will include a suitably encoded public cryptographic key.

1.1. Rationale and Intent

   These identifiers are expected to be used at the existing IPv6
   Application Programming Interfaces (API) and application protocols
   between consenting hosts. They may be defined and used in different
   contexts, suitable for different overlay protocols. Examples of
   these include Host Identity Tags (HIT) in the Host Identity Protocol
   (HIP) [I-D.ietf-hip-rfc5201-bis] and Temporary Mobile Identifiers
   (TMI) for Mobile IPv6 Privacy Extension [PRIVACYTEXT].

   As these identifiers are expected to be used along with IPv6
   addresses at both applications and APIs, co-ordination is desired to
   make sure that an ORCHID is not inappropriately taken for a vanilla
   IPv6 address and vice versa. In practice, allocation of a separate
   prefix for ORCHIDs seems to suffice, making them compatible with IPv6
   addresses at the upper layers while simultaneously making it trivial
   to prevent their usage at the IP layer.

   While being technically possible to use ORCHIDs between consenting
   hosts without any co-ordination with the IETF and the IANA, the
   authors would consider such practice potentially dangerous. A
   specific danger would be realised if the IETF community later decided
   to use the ORCHID prefix for some different purpose. In that case,
   hosts using the ORCHID prefix would be, for practical purposes,
   unable to use the prefix for the other new purpose. That would lead
   to partial balkanisation of the Internet, similar to what has
   happened as a result of historical hijackings of non-RFC 1918
   [RFC1918] IPv4 addresses for private use.

   The whole need for the proposed allocation grows from the desire to
   be able to use ORCHIDs with existing applications and APIs. This
   desire leads to the potential conflict, mentioned above. Resolving
   the conflict requires the proposed allocation.

   One can argue that the desire to use these kinds of identifiers via
   existing APIs is architecturally wrong, and there is some truth in
   that argument. Indeed, it would be more desirable to introduce a new

API and update all applications to use identifiers, rather than
locators, via that new API. That is exactly what we expect to happen
in the long run.

However, given the current state of the Internet, we do not consider
it viable to introduce any changes that, at once, require
applications to be rewritten and host stacks to be updated. Rather
than that, we believe in piece-wise architectural changes that
require only one of the existing assets to be touched. ORCHIDs are
designed to address this situation: to allow people to experiment
with protocol stack extensions, such as secure overlay routing, HIP,
or Mobile IP privacy extensions, without requiring them to update
their applications. The goal is to facilitate large-scale
experiments with minimum user effort.

For example, there already exists, at the time of this writing, HIP
implementations that run fully in user space, using the operating
system to divert a certain part of the IPv6 address space to a user
level daemon for HIP processing. In practical terms, these
implementations are already using a certain IPv6 prefix for
differentiating HIP identifiers from IPv6 addresses, allowing them
both to be used by the existing applications via the existing APIs.

The Overlay Routable Cryptographic Hash Identifiers originally
defined in [RFC4843] lacked a mechanism for cryptographic algorithm
agility. The updated ORCHID format specified in this document
removes this limitation by encoding in the identifier itself an index
to the suite of cryptographic algorithms in use.

Because the updated ORCHIDv2 format is not backward compatible with
the earlier one, IANA is requested to allocate a new 28-bit prefix
out of the IANA IPv6 Special Purpose Address Block, namely 2001:
0000::/23, as per [RFC4773]. The prefix that was temporarily
allocated for the experimental ORCHID is to be returned to IANA in
2014 [RFC4843].

1.2. ORCHID Properties

ORCHIDs are designed to have the following properties:

o Statistical uniqueness; also see Section 4

o Secure binding to the input parameters used in their generation
   (i.e., the context identifier and a bitstring).

o Aggregation under a single IPv6 prefix. Note that this is only
   needed due to the co-ordination need as indicated above. Without
   such co-ordination need, the ORCHID namespace could potentially be

completely flat.

o Non-routability at the IP layer, by design.

o Routability at some overlay layer, making them, from an
   application point of view, semantically similar to IPv6 addresses.

As mentioned above, ORCHIDs are intended to be generated and used in
different contexts, as suitable for different mechanisms and
protocols. The context identifier is meant to be used to
differentiate between the different contexts; see Section 4 for a
discussion of the related API and kernel level implementation issues,
and Section 5 for the design choices explaining why the context
identifiers are used.

1.3. Expected use of ORCHIDs

Examples of identifiers and protocols that are expected to adopt the
ORCHID format include Host Identity Tags (HIT) in the Host Identity
Protocol [I-D.ietf-hip-rfc5201-bis] and the Temporary Mobile
Identifiers (TMI) in the Simple Privacy Extension for Mobile IPv6
[PRIVACYTEXT]. The format is designed to be extensible to allow
other experimental proposals to share the same namespace.

1.4. Action Plan

This document requests IANA to allocate a prefix out of the IPv6
addressing space for Overlay Routable Cryptographic Hash Identifiers.

1.5. Terminology

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

2. Cryptographic Hash Identifier Construction

An ORCHID is generated using the ORCHID Generation Algorithm (OGA)
below. The algorithm takes a bitstring and a context identifier as
input and produces an ORCHID as output. The hash function used in
the ORCHID Generation Algorithm is defined for each OGA identifier by
the specification for the respective usage context (e.g., HIPv2).

```
Input      :=  any bitstring
OGA ID     :=  4-bits Orchid Generation Algorithm identifier
Hash Input :=  Context ID | Input
Hash       :=  Hash_function( Hash Input )
ORCHID     :=  Prefix | Encode_96( Hash )
```

where:

| : Denotes concatenation of bitstrings

Input   : A bitstring that is unique or statistically unique
                  within a given context. The bitstring is intended
                  to be associated with the to-be-created ORCHID in
                  the given context.

Context ID   : A randomly generated value defining the expected
                  usage context for the particular ORCHID and the
                  hash function to be used for generation of ORCHIDs
                  in this context. These values are allocated out of
                  the namespace introduced for CGA Type Tags; see RFC
                  3972 and
                  http://www.iana.org/assignments/cga-message-types.

OGA ID   : A 4-bit long identifier for the Hash_function
                in use within the specific usage context.

Hash_function   : The one-way hash function (i.e., hash function
                  with pre-image resistance and second pre-image
                  resistance) to be used as identified by the
                  value for the OGA ID according document
                  defining the context usage identified by the
                  Context ID. For example, the version 2 of the
                  HIP specification defines SHA1 [RFC3174] as
                  the hash function to be used to generate
                  ORCHIDv2 used in the HIPv2 protocol when the
                  OGA ID is 3 [I-D.ietf-hip-rfc5201-bis].

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

Prefix   : A constant 28-bit-long bitstring value
                  (IANA TBD 2001:11::/28 ?).

To form an ORCHID, two pieces of input data are needed. The first
piece can be any bitstring, but is typically expected to contain a
public cryptographic key and some other data. The second piece is a

context identifier, which is a 128-bit-long datum, allocated as
specified in Section 7. Each specific experiment (such as HIP HITs
or MIP6 TMIs) is expected to allocate their own, specific context
identifier.

The input bitstring and context identifier are concatenated to form
an input datum, which is then fed to the cryptographic hash function
to be used for the value of the OGA identifier according to the
document defining the context usage identified by the Context ID.
The result of the hash function is processed by an encoding function,
resulting in a 96-bit-long value. This value is prepended with the
concatenation of the 28-bit ORCHID prefix and the 4-bit OGA ID. The
result is the ORCHID, a 128-bit-long bitstring that can be used at
the IPv6 APIs in hosts participating to the particular experiment.

The ORCHID prefix is allocated under the IPv6 global unicast address
block. Hence, ORCHIDs are indistinguishable from IPv6 global unicast
addresses. However, it should be noted that ORCHIDs do not conform
with the IPv6 global unicast address format defined in Section 2.5.4
of [RFC4291] since they do not have a 64-bit Interface ID formatted
as described in Section 2.5.1. of [RFC4291].

3. Routing Considerations

ORCHIDs are designed to serve as location independent endpoint-
identifiers rather than IP-layer locators. Therefore, routers MAY be
configured not to forward any packets containing an ORCHID as a
source or a destination address. If the destination address is an
ORCHID but the source address is a valid unicast source address,
routers MAY be configured to generate an ICMP Destination
Unreachable, Administratively Prohibited message.

Due to the experimental nature of ORCHIDs, router software MUST NOT
include any special handling code for ORCHIDs. In other words, the
non-routability property of ORCHIDs, if implemented, MUST be
implemented via configuration and NOT by hardwired software code. At
this time, it is RECOMMENDED that the default router configuration
not handle ORCHIDs in any special way. In other words, there is no
need to touch existing or new routers due to this experiment. If
such a reason should later appear, for example, due to a faulty
implementation leaking ORCHIDs to the IP layer, the prefix can be and
should be blocked by a simple configuration rule.

3.1. Overlay Routing

As mentioned multiple times, ORCHIDs are designed to be non-routable
at the IP layer. However, there are multiple ongoing research
efforts for creating various overlay routing and resolution

mechanisms for flat identifiers. For example, the Host Identity
Indirection Infrastructure (Hi3) [Hi3] and Node Identity
Internetworking Architecture (NodeID) [NodeID] proposals, outline
ways for using a Distributed Hash Table to forward HIP packets based
on the Host Identity Tag.

What is common to the various research proposals is that they create
a new kind of resolution or routing infrastructure on top of the
existing Internet routing structure. In practical terms, they allow
delivery of packets based on flat, non-routable identifiers,
utilising information stored in a distributed database. Usually, the
database used is based on Distributed Hash Tables. This effectively
creates a new routing network on top of the existing IP-based routing
network, capable of routing packets that are not addressed by IP
addresses but some other kind of identifiers.

Typical benefits from overlay routing include location independence,
more scalable multicast, anycast, and multihoming support than in IP,
and better DoS resistance than in the vanilla Internet. The main
drawback is typically an order of magnitude of slower performance,
caused by an easily largish number of extra look-up or forwarding
steps needed. Consequently, in most practical cases, the overlay
routing system is used only during initial protocol state set-up (cf.
TCP handshake), after which the communicating endpoints exchange
packets directly with IP, bypassing the overlay network.

The net result of the typical overlay routing approaches is a
communication service whose basic functionality is comparable to that
provided by classical IP but provides considerably better resilience
that vanilla IP in dynamic networking environments. Some experiments
also introduce additional functionality, such as enhanced security or
ability to effectively route through several IP addressing domains.

The authors expect ORCHIDs to become fully routable, via one or more
overlay systems, before the end of the experiment.

4. Collision Considerations

As noted above, the aim is that ORCHIDs are globally unique in a
statistical sense. That is, given the ORCHID referring to a given
entity, the probability of the same ORCHID being used to refer to
another entity elsewhere in the Internet must be sufficiently low so
that it can be ignored for most practical purposes. We believe that
the presented design meets this goal; see Section 5.

Consider next the very rare case that some ORCHID happens to refer to
two different entities at the same time, at two different locations
in the Internet. Even in this case, the probability of this fact

becoming visible (and therefore a matter of consideration) at any
single location in the Internet is negligible. For the vast majority
of cases, the two simultaneous uses of the ORCHID will never cross
each other. However, while rare, such collisions are still possible.
This section gives reasonable guidelines on how to mitigate the
consequences in the case that such a collision happens.

As mentioned above, ORCHIDs are expected to be used at the legacy
IPv6 APIs between consenting hosts. The context ID is intended to
differentiate between the various experiments, or contexts, sharing
the ORCHID namespace. However, the context ID is not present in the
ORCHID itself, but only in front of the input bitstring as an input
to the hash function. While this may lead to certain implementation-
related complications, we believe that the trade-off of allowing the
hash result part of an ORCHID being longer more than pays off the
cost.

Because ORCHIDs are not routable at the IP layer, in order to send
packets using ORCHIDs at the API level, the sending host must have
additional overlay state within the stack to determine which
parameters (e.g., what locators) to use in the outgoing packet. An
underlying assumption here, and a matter of fact in the proposals
that the authors are aware of, is that there is an overlay protocol
for setting up and maintaining this additional state. It is assumed
that the state-set-up protocol carries the input bitstring, and that
the resulting ORCHID-related state in the stack can be associated
back with the appropriate context and state-set-up protocol.

Even though ORCHID collisions are expected to be extremely rare, two
kinds of collisions may still happen. First, it is possible that two
different input bitstrings within the same context may map to the
same ORCHID. In this case, the state-set-up mechanism is expected to
resolve the conflict, for example, by indicating to the peer that the
ORCHID in question is already in use.

A second type of collision may happen if two input bitstrings, used
in different usage contexts, map to the same ORCHID. In this case,
the main confusion is about which context to use. In order to
prevent these types of collisions, it is RECOMMENDED that
implementations that simultaneously support multiple different
contexts maintain a node-wide unified database of known ORCHIDs, and
indicate a conflict if any of the mechanisms attempt to register an
ORCHID that is already in use. For example, if a given ORCHID is
already being used as a HIT in HIP, it cannot simultaneously be used
as a TMI in Mobile IP. Instead, if Mobile IP attempts to use the
ORCHID, it will be notified (by the kernel) that the ORCHID in
question is already in use.

5. Design Choices

   The design of this namespace faces two competing forces:

   o As many bits as possible should be preserved for the hash result.

   o It should be possible to share the namespace between multiple
      mechanisms.

   The desire to have a long hash result requires that the prefix be as
   short as possible, and use few (if any) bits for additional encoding.
   The present design takes this desire to the maximum: all the bits
   beyond the prefix and the ORCHID generation algorithm identifier are
   used as hash output. This leaves no bits in the ORCHID itself
   available for identifying the context, however the 4 bits used to
   encode the ORCHID generation algorithm identifier provides
   cryptographich agility with respect to the hash function in use for a
   given context; see Section 6.

   The desire to allow multiple mechanisms to share the namespace has
   been resolved by including the context identifier in the hash-
   function input. While this does not allow the mechanism to be
   directly inferred from a ORCHID, it allows one to verify that a given
   input bitstring and ORCHID belong to a given context, with high-
   probability; but also see Section 6.

6. Security Considerations

   ORCHIDs are designed to be securely bound to the Context ID and the
   bitstring used as the input parameters during their generation. To
   provide this property, the ORCHID generation algorithm relies on the
   second-preimage resistance (a.k.a. one-way) property of the hash
   function used in the generation [RFC4270]. To have this property and
   to avoid collisions, it is important that the allocated prefix is as
   short as possible, leaving as many bits as possible for the hash
   output.

   For a given Context ID, all mechanisms using ORCHIDs MUST use exactly
   the same mechanism for generating an ORCHID from the input bitstring.
   Allowing different mechanisms, without explicitly encoding the
   mechanism in the Context ID or the ORCHID itself, would allow so-
   called bidding-down attacks. That is, if multiple different hash
   functions were allowed to construct ORCHIDs valid for the same
   Context ID, and if one of the hash functions became insecure, that
   would allow attacks against even those ORCHIDs valid for the same
   Context ID that had been constructed using the other, still secure
   hash functions.

An identifier for the hash function to be used for the ORCHID
generation is encoded in the ORCHID itself, while the semantic for
the values taken by this identifier are defined separately for each
Context ID. Therefore, the present design allows to use different
hash functions to be used per given Context ID for constructing
ORCHIDs from input bitstrings. If more secure hash functions are
later needed, newer values for the ORCHID generation algorithm can be
defined for the given Context ID.

In order to preserve a low enough probability of collisions (see
Section 4), each method MUST utilize a mechanism that makes sure that
the distinct input bitstrings are either unique or statistically
unique within that context. There are several possible methods to
ensure this; for example, one can include into the input bitstring a
globally maintained counter value, a pseudo-random number of
sufficient entropy (minimum 96 bits), or a randomly generated public
cryptographic key. The Context ID makes sure that input bitstrings
from different contexts never overlap. These together make sure that
the probability of collisions is determined only by the probability
of natural collisions in the hash space and is not increased by a
possibility of colliding input bitstrings.

7. IANA Considerations

Because the updated ORCHIDv2 format is not backward compatible with
the earlier one, IANA is requested to allocate a new 28-bit prefix
out of the IANA IPv6 Special Purpose Address Block, namely 2001:
0000::/23, as per [RFC4773]. The prefix that was temporarily
allocated for the experimental ORCHID is to be returned to IANA in
2014 [RFC4843].

The Context Identifier (or Context ID) is a randomly generated value
defining the usage context of an ORCHID and the hash function to be
used for generation of ORCHIDs in this context. This document
defines no specific value. The Context ID shares the name space
introduced for CGA Type Tags. Hence, defining new values follows the
rules of Section 8 of [RFC3972], i.e., First Come First Served.

8. Contributors

Pekka Nikander (pekka.nikander@nomadiclab.com) co-authored an
earlier, experimental version of this specification [RFC4843].

9. Acknowledgments

Special thanks to Geoff Huston for his sharp but constructive
critique during the development of this memo. Tom Henderson helped
to clarify a number of issues. This document has also been improved

by reviews, comments, and discussions originating from the IPv6,
Internet Area, and IETF communities.

10. References

10.1. Normative references

[RFC2119]   Bradner, S., "Key words for use in RFCs
                             to Indicate Requirement Levels", BCP 14,
                             RFC 2119, March 1997.

[RFC3972]   Aura, T., "Cryptographically Generated
                             Addresses (CGA)", RFC 3972, March 2005.

10.2. Informative references

[Hi3]   Nikander, P., Arkko, J., and B. Ohlman,
                             "Host Identity Indirection Infrastructure
                             (Hi3)", November 2004.

[I-D.ietf-hip-rfc5201-bis]   Moskowitz, R., Heer, T., Jokela, P., and
                             T. Henderson, "Host Identity Protocol
                             Version 2 (HIPv2)",
                             draft-ietf-hip-rfc5201-bis-09 (work in
                             progress), July 2012.

[NodeID]   Ahlgren, B., Arkko, J., Eggert, L., and
                             J. Rajahalme, "A Node Identity
                             Internetworking Architecture (NodeID)",
                             April 2006.

[PRIVACYTEXT]   Dupont, F., "A Simple Privacy Extension
                             for Mobile IPv6", Work in Progress,
                             July 2006.

[RFC1918]   Rekhter, Y., Moskowitz, R., Karrenberg,
                             D., Groot, G., and E. Lear, "Address
                             Allocation for Private Internets", BCP 5,
                             RFC 1918, February 1996.

[RFC3174]   Eastlake, D. and P. Jones, "US Secure
                             Hash Algorithm 1 (SHA1)", RFC 3174,
                             September 2001.

[RFC4270]   Hoffman, P. and B. Schneier, "Attacks on
                             Cryptographic Hashes in Internet
                             Protocols", RFC 4270, November 2005.

    [RFC4291]   Hinden, R. and S. Deering, "IP Version 6
                                    Addressing Architecture", RFC 4291,
                                    February 2006.

    [RFC4773]   Huston, G., "Administration of the IANA
                                    Special Purpose IPv6 Address Block",
                                    RFC 4773, December 2006.

    [RFC4843]   Nikander, P., Laganier, J., and F.
                                    Dupont, "An IPv6 Prefix for Overlay
                                    Routable Cryptographic Hash Identifiers
                                    (ORCHID)", RFC 4843, April 2007.

Appendix A. Changes from RFC 4843

o Updated HIP references to revised HIP specifications.

o The Overlay Routable Cryptographic Hash Identifiers originally
   defined in [RFC4843] lacked a mechanism for cryptographic
   algorithm agility. The updated ORCHID format specified in this
   document removes this limitation by encoding in the identifier
   itself an index to the suite of cryptographic algorithms in use.

Authors' Addresses

    Julien Laganier
    Juniper Networks
    1094 North Mathilda Avenue
    Sunnyvale, CA 94089
    USA

    Phone: +1 408 936 0385
    EMail: julien.ietf@gmail.com

    Francis Dupont
    Internet Systems Consortium

    EMail: fdupont@isc.org