HIP Working Group J. Laganier
Internet-Draft LIP / Sun Microsystems
Expires: April 18, 2005 L. Eggert
NEC
October 18, 2004

Host Identity Protocol (HIP) Rendezvous Extensions
draft-ietf-hip-rvs-00

Status of this Memo

Copyright Notice

Abstract

This document discusses rendezvous extensions for the Host Identity Protocol (HIP). Rendezvous mechanisms extend HIP for communication with HIP Rendezvous Servers. Rendezvous Servers improve operation when HIP nodes are multi-homed or mobile. The first part of his document motivates the need for rendezvous mechanisms; the second part describes the protocol extensions in detail.

Table of Contents

1. Introduction 3
2. Terminology 4
3. Communication Between HIP Nodes 4
4. Communication Between Mobile or Multi-Homed HIP Nodes 6
4.1 Mobility and Multi-Homing with DNS Updates 6
4.2 Mobility and Multi-Homing with Rendezvous Servers 8
5. HIP Extensions for Rendezvous Servers 10
5.1 Additional RVS_CAPABLE Control Field 10
5.2 Additional HIP Parameters 10
5.2.1 RVA_REQUEST Parameter Format and Processing 10
5.2.2 RVA_REPLY Parameter Format and Processing 12
5.2.3 RVA_HMAC Parameter Format and Processing 12
5.2.4 FROM Parameter Format and Processing 13
5.2.5 TO Parameter Format and Processing 14
5.2.6 VIA_RVS Parameter Format and Processing 15
5.3 Use of Existing HIP Messages and Parameters 16
5.3.1 ECHO_REQUEST and ECHO_REPLY Parameters 16
5.3.2 REA Parameter 16
6. Diagram Notation 17
7. Establishing Rendezvous Associations 17
8. Establishing HIP Associations via Rendezvous Servers 20
8.1 Sending a Redirect in Reply to I1 20
8.2 Passing I1 onto an ESP SA 21
8.3 Rewriting I1 Destination IP Address 22
8.4 Rewriting I1 Source and Destination IP Addresses 23
8.5 Rewriting I1 and R1 Source and Destination IP Addresses . 24
8.6 Cascading Rendezvous Servers 26
8.7 Implication on the HIP integrity checks 27
8.7.1 Checksum 27
8.7.2 HMAC and SIGNATURE 27
8.7.3 Example 28
9. Security Considerations 28
10. IANA Considerations 29
11. Acknowledgments 29
12. References 30
12.1 Normative References 30
12.2 Informative References 31
Authors' Addresses 31
A. Document Revision History 32
Intellectual Property and Copyright Statements 33

1. Introduction

The current Internet uses two global namespaces: domain names and IP addresses. The Domain Name System (DNS) provides a two-way lookup service between the two [1]. Domain names are symbolic identifiers for sets of IP addresses.

IP addresses have two uses. First, they are topological locators for network attachment points. Second, they act as names for the attached network interfaces. Saltzer [11] discusses these naming concepts in detail.

Routing and other network-layer mechanisms are based on the locator aspects of IP addresses. Transport-layer protocols and mechanisms typically use IP addresses in their role as names for communication endpoints.

This dual use of IP addresses limits the flexibility of the Internet architecture. The need to avoid readdressing in order to maintain existing transport-layer connections complicates advanced functionality, such as mobility, multi-homing, or network composition.

The Host Identity Protocol (HIP) architecture [2] defines a new third namespace. The Host Identity namespace decouples the name and locator roles currently filled by IP addresses. Instead of mapping domain names directly into IP addresses, HIP maps domain names into Host Identities, and Host Identities into IP addresses. Transport-layer mechanisms operate on Host Identities instead of using IP addresses as endpoint names. Network-layer mechanisms continue to use IP addresses as pure locators.

Without HIP, nodes establish transport-layer connections by first looking up the fully-qualified domain name (FQDN) of a peer in the DNS. A successful DNS lookup returns the peer's IP addresses. A node uses one of the returned IP addresses to initiate transport-layer communication with a peer node.

HIP nodes will also look up the domain name of desired peers in the DNS, as specified in the HIP DNS Extensions[3]. When a successful lookup includes a peer's Host Identities, HIP nodes perform a HIP Base Exchange before establishing transport-layer connections. The HIP Base Exchange authenticates the end hosts and can bootstrap encryption of the subsequent communication with IPsec [12]. The HIP specification [4] discusses the details of the Base Exchange and the related protocol exchanges.

After the Base Exchange, HIP nodes use Host Identities instead of IP

addresses for transport-layer connections with a peer. The HIP layer in the network stack internally translates Host Identities (HI) into network-layer IP addresses. This additional mapping between Host Identities and IP addresses (HI->IP) is logically separate from the first mapping between fully-qualified domain names and Host Identities (FQDN->HI).

For application and transport-layer compatibility, the FQDN->HI mapping must remain in the DNS. However, the HI->IP mapping is internal to the HIP layer and may be performed in a number of ways. Different lookup mechanism may support communication between two mobile or multi-homed HIP nodes better [5].

2. Terminology

Rendezvous Server (RVS): A HIP enabled node which relays incoming HIP I1 packets to the owner of the receiver HIT contained in the I1 header. A RVS may also relay back an R1 to an opportunistic Initiator.

Rendezvous Association (RVA): A lightweight HIP association established between a HIP node and its RVS. The associated state doesn't require communication to be maintained and contains the peer's HIT, two symmetric integrity keys, and the IP addresses of both nodes.

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 RFC 2119 [6].

3. Communication Between HIP Nodes

In the current Internet, the DNS provides a FQDN->IP mapping. With HIP, it must continue to provide a mapping based on domain names. This allows transport-layer connections to bind to Host Identities instead of IP addresses transparently.

Instead of mapping domain names directly into IP addresses (FQDN->IP), with HIP the DNS maps them to Host Identities (FQDN->HI). In a second step, another lookup that is internal to the HIP-layer translates the Host Identities into IP addresses for network-layer delivery (HI->IP).

Several alternative approaches are possible for maintaining the HI->IP information. The DNS can maintain this mapping along with the FQDN->HI mapping. Alternatively, a database separate from the DNS can manage this information. This section discusses the different approaches and their implications on communication between two HIP

nodes.

The HIP architecture, protocol and DNS extensions specifications suggest storing Host Identities along with a node's IP addresses in the DNS [3][2][4]. The index for both tables will be domain names. Logically, the DNS will thus contain two separate mappings: FQDN->HI and FQDN->IP.

Figure 1 shows the lookup steps and HIP Base Exchange when a node's Host Identities are stored alongside its IP addresses. In step #1, the Initiator I performs a DNS lookup on R's domain name FQDN(R). The DNS server responds with both R's Host Identities HI(R) and its IP addresses IP(R) in step #2 (Details can be found in [4]).

The Initiator I uses both pieces of information to perform the HIP Base Exchange with R in step #3. (The details of the Base Exchange, specified in [4], are not relevant to this discussion and will thus be omitted.)

```
                    #1 FQDN(R)      +----------+
              +-------------------->|   DNS    |
              | +-------------------|          |
              | |  #2 HI(R), IP(R)  | FQDN->HI |
              | |                   | FQDN->IP |
              | |                   +----------+
              | V
            +-----+       #3 HIP Base Exchange      +-----+
            |     |-------------------------------->|     |
            |  I  |<--------------------------------|  R  |
            |     |-------------------------------->|     |
            |     |<--------------------------------|     |
            +-----+                                 +-----+
```

Figure 1: HIP Lookup and Base Exchange

Note that the DNS does not currently store the HI->IP mapping directly. Instead, a DNS lookup on a domain name returns both its FQDN->HI and FQDN->IP entries. The HIP stack then implicitly constructs the HI->IP mapping based on the HI and IP information returned by the DNS lookup. In the example in Figure 1, the FQDN(R) lookup in step #1 returns both HI(R) and IP(R) in step #2. HIP implicitly constructs the HI(R)->IP(R) mapping based on the assumption that HI(R) is reachable at IP(R).

One disadvantage of this approach is that a node's domain name is required to obtain both its Host Identities and its IP addresses. Even if a HIP node already knows the Host Identity of a HIP peer through other means, it cannot currently obtain the peer's IP

addresses through the DNS. The DNS does not maintain an explicit HI->IP table, but instead indexes Host Identities only by domain names.

A reverse HI->FQDN DNS mapping could address this limitation. HIP nodes would then look up a HIP peer's domain name through its Host Identity. They would then use the returned domain name to find the peer's IP addresses in a second lookup. However, the DNS may not be structurally suited to maintain the reverse HIP->FQDN mapping. As the main Internet-wide database, the DNS is already being overloaded with functionality that might be better handled with new mechanisms [13]. Finally, the additional reverse lookup would increase the latency of the HIP Base Exchange.

4. Communication Between Mobile or Multi-Homed HIP Nodes

HIP decouples domain names from IP addresses. Because transport protocols bind to Host Identities, they remain unaware if the set of IP addresses associated with a Host Identity changes. This change can have various reasons, including, but not limited to, mobility and multi-homing.

Proposed extensions for mobility and multi-homing [5] allow a HIP node to notify its peers about changes in its set of IP addresses. These extensions require an established HIP association between two nodes, i.e., a completed HIP Base Exchange.

In addition to notifying its current peers about changes in its IP addresses, a HIP node must also update its HI->IP mapping in response to IP address changes. Otherwise, HIP Base Exchanges from new peers could fail because they try to contact the node at an IP address it is no longer reachable at.

4.1 Mobility and Multi-Homing with DNS Updates

If the DNS indirectly maintains the HI->IP mapping in a FQDN->IP table, nodes can dynamically update their DNS entry in a secure fashion [7][8]. The DNS server maintaining the information will then sign and distribute the updated zone.

```
              #2 FQDN(R)     +----------+
       +--------------------->|   DNS    |
       | +-------------------|          |<------+
       | |  #3 HI(R), IP(R)  | FQDN->HI |       | #1 Update
       | |                   | FQDN->IP |       |    FQDN(R)->IP(R)
       | |                   +----------+       |    whenever IP(R)
       | V                                      |    changes.
     +-----+       #4 HIP Base Exchange      +-----+
     |     |-------------------------------->|     |
     |  I  |<--------------------------------|  R  |
     |     |-------------------------------->|     |
     |     |<--------------------------------|     |
     +-----+                                 +-----+
```

Figure 2: HIP Lookup and Base Exchange with DNS Updates

Figure 2 shows an example of this scenario. In step #1, R registers its FQDN(R)->IP(R) entry in the DNS. It will dynamically update the DNS entry whenever its IP addresses IP(R) change. Because the DNS always contains R's current IP addresses, node I can perform a HIP Base Exchange with R at its new IP address (steps #2-4).

One drawback of using dynamic DNS updates in this way is the cost of updating secure zones. Re-signing an entire zone whenever the IP addresses of one entry change places a high cost on the DNS server. Using dynamic DNS to update HI->IP mappings may thus not be appropriate when changes of IP addresses are frequent.

A simple, operational change could help limit the costs of frequent DNS updates. Instead of recomputing a zone after each dynamic update, a DNS server could aggregate the modifications and only perform zone updates periodically. The disadvantage of this approach is that HIP nodes may be unreachable until the DNS server distributes the updated zone.

Another concern with using the DNS to support HIP node mobility is the propagation time of updated DNS entries. DNS servers frequently cache DNS responses to reduce the load on the primary servers. During the time-to-live associated with a DNS response, DNS servers may answer additional requests for the same DNS entry from their local caches instead of contacting the primary servers. Thus, even after a HIP node updates its DNS entry, the DNS can still serve the old entry until the cached responses expire. This can lead to communication problems, because peers may try to contact a HIP node at an IP address it is no longer reachable at.

4.2 Mobility and Multi-Homing with Rendezvous Servers

The HIP architecture tries to greatly reduce the frequency of Dynamic DNS updates by introducing Rendezvous Servers [2]. Instead of registering its current set of IP addresses in its HI->IP entry in the DNS, a HIP node may instead register the IP addresses of its Rendezvous Servers. Because the IP addresses of Rendezvous Servers are assumed to change only infrequently, this approach can significantly reduce the load on DNS servers.

Rendezvous Servers maintain a mapping between the Host Identities of HIP nodes for which they provide service and the node's current IP addresses. HIP nodes must notify their Rendezvous Servers about any changes in their IP addresses. This approach effectively relocates the HI->IP information - and the burden of keeping it current - from the DNS to the Rendezvous Servers. This can reduce update costs under the assumption that Rendezvous Servers provide more efficient ways of maintaining HI->IP tables.

When a packet destined for one of its HIP nodes arrives at a Rendezvous Server, it relays the packet to one of the HIP node's current IP addresses. Due to the specifics of the HIP, only the first packet of a HIP Base Exchange will require such relaying [2]. Subsequent packet of the HIP Base Exchange and all further data packets will directly flow between the HIP nodes, bypassing the Rendezvous Server.

```
               #3 FQDN(R)      +----------+ #2 Register IP(RVS) in
        +--------------------->|   DNS    |    FQDN(R)->IP(RVS).
        | +--------------------|          |<------------------+
        | |  #4 HI(R), IP(RVS) | FQDN->HI |                   |
        | |                    | FQDN->IP |                   |
        | |                    +----------+                   |
        | |                                                   |
        | |                   #1 Update IP(R) in HI(R)->IP(R) |
        | |        +--------+    whenever IP(R) changes.      |
        | |        |  RVS   |<------------------------------+ |
        | |        |        |                               | |
        | V     +->| HI->IP |--+                            | |
      +-----+   |  +--------+  |                          +-----+
      |     |---+              +------------------------->|     |
      |  I  |    #5 First Message of HIP Base Exchange    |  R  |
      |     |                                             |     |
      |     |<--------------------------------------------|     |
      |     |-------------------------------------------->|     |
      |     |<--------------------------------------------|     |
      +-----+       #6 Remainder of HIP Base Exchange     +-----+
```

Figure 3: HIP Lookup and Base Exchange with Rendezvous Server

Figure 3 shows a HIP lookup and Base Exchange involving a Rendezvous Server. Here, HIP node R is using Rendezvous Server RVS. In step #1, it updates RVS with its current IP addresses IP(R). Then, in step #2, R registers the Rendezvous Server's IP addresses IP(RVS) in its FQDN(R)->IP(RVS) DNS entry.

In step #3, a second HIP node I issues a DNS lookup on FQDN(R) to obtain R's Host Identities HI(R) and IP addresses. The lookup returns R's Host Identities HI(R) in step #4. The DNS reply also includes the IP addresses of the Rendezvous Server IP(RVS) (instead of IP(R), because R's current addresses are unknown to the DNS.)

In step #5, node I initiates the HIP Base Exchange. It addresses the first packet of the HIP Base Exchange to IP(RVS). Upon receipt, the Rendezvous Server relays the packet to one of R's current IP addresses IP(R). The remainder of the HIP Base Exchange then occurs directly between I and R in step #6.

When Rendezvous Servers maintain the HI->IP information, they may support more efficient update operations compared to dynamic DNS updates (Section 4.1). Unlike the DNS, Rendezvous Servers do not provide a lookup service. Instead, they use the HI->IP information to actively relay traffic between HIP nodes.

This approach changes the role of the IP addresses stored in a DNS

entry. Traditionally, nodes were directly reachable at the IP addresses listed in their DNS entry. HIP Rendezvous Server change this basic property by replacing the IP addresses of their client nodes in the DNS with their own. The IP addresses in a DNS entry hence no longer directly designate interfaces of an endpoint. Instead, they identify interfaces of a node that can relay packets to the endpoint.

5. HIP Extensions for Rendezvous Servers

The following sections describe HIP extensions for communication with Rendezvous Servers. These extensions allow:

- A HIP Rendezvous Server to advertise its RVS capabilities to its correspondents.
- A HIP node to create a Rendezvous Association (RVA) with its Rendezvous Server, i.e., to register its current set of IP address(es).
- Two HIP nodes to establish a HIP Association (HA) between them via one or more Rendezvous Server.

5.1 Additional RVS_CAPABLE Control Field

RVS mechanisms make use of a new Control Fields in the HIP Control Field: the RVS_CAPABLE Control Field.

The RVS_CAPABLE Control Field ("R") allows a Rendezvous Server to advertise its rendezvous capabilities to the HIP nodes it associates with.

5.2 Additional HIP Parameters

5.2.1 RVA_REQUEST Parameter Format and Processing

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Lifetime                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           RVA Type #1         |           RVA Type #2         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           RVA Type #n         |             padding           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
Type         100
Length       Length in octets, excluding Type, Length and Padding
Lifetime     This field encode, the desired RVA validity time.
RVA Type     This field encode, in order of preference, the
             preferred rendezvous service types.
```

The following RVA Types are defined:

Type number	RVA Type
0	Reserved by IANA
1	I1_REWRITE_DST
2	I1_REWRITE_SRCDST
3	I1R1_REWRITE_SRCDST
4	I1_RELAY_ESP
5	I1R1_RELAY_ESP
6	REDIRECT
6-200	Reserved by IANA
201-255	Reserved by IANA for private use

When a Rendezvous Association of type I1_* is established between a HIP RVS and its peer, the RVS will relay to the peer all inbound I1s whose Responder HIT match those of the peer. The peer will then reply with a R1 sent directly to the Initiator, without further assistance from the RVS.

When a Rendezvous Association of type I1R1_* is established between a HIP RVS and its peer, the RVS will relay to the peer all inbound I1s whose Responder HIT match those of the peer. The peer will then reply with a R1 sent to the Initiator via the RVS, which will relay it to the Initiator. The Initiator will then reply directly to the Responder by sending an I2, without further assistance from the RVS.

A RVS relays packet by either rewriting IP addresses in the IP header, or alternatively, if a HIP association is present, by forwarding it into the ESP SA associated with the HIP Association.

If the RVA is of type *_REWRITE_*, the IP addresses are rewritten by the RVS. If the RVA type is I1_REWRITE_DST, only the destination IP address of a relayed I1 is rewritten. On the contrary, if the RVA type *_REWRITE_SRCDST, both the source and destination IP addresses are rewritten. In the case of a *_REWRITE_SRCDST, the RVS will need to suffix the HIP header with a FROM parameter preserving the original source IP address of the relayed packet. This FROM, as well as the whole HIP header, is integrity protected by an RVA_HMAC parameter which contains a keyed-HMAC computed over the HIP packet, similarly to what the HMAC parameter already does.

5.2.2 RVA_REPLY Parameter Format and Processing

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Lifetime                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           RVA Type #1         |           RVA Type #2         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           RVA Type #n         |             padding           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
Type          102
Length        Length in octets, excluding Type, Length and Padding
Lifetime      This field encode the offered RVA validity time
RVA Type      This field encode, in order of preference, the
              preferred rendezvous service types (the same
              type values than RVA_REQUEST parameter are used).
```

5.2.3 RVA_HMAC Parameter Format and Processing

The RVA_HMAC is an OPTIONAL parameter whose only difference with the HMAC parameter defined in [4] is the Type code:

Type 65320
Length 20
HMAC 160 low order bits of a HMAC keyed with the appropriate HIP integrity keys (HIP_lg or HIP_gl) of the corresponding Rendezvous Association or HIP Association. This HMAC is computed over the HIP packet excluding RVA_HMAC and any other following parameter. The checksum field MUST be set to zero and the HIP header length in the HIP common header MUST be calculated not to cover any excluded parameter when the Authenticator field is calculated.

To allow a HIP node and any of its RVS to verify the integrity of packets flowing between them, both use an RVA_HMAC parameter keyed with a HMAC of HIP_lg and HIP_gl integrity keys. One RVA_HMAC SHOULD be present on every packets flowing between a HIP node and any of its RVS and MUST be present when FROM and TO parameters are processed.

On the receiving side, when an RVA_HMAC is validated, it SHOULD be removed from the packet and if so, packet length and checksum MUST be recomputed accordingly.

5.2.4 FROM Parameter Format and Processing

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                            Address                            |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type 65100 (under signature) or 65300 (after signature)
Length 16
Address An IPv6 address or an IPv4-in-IPv6 format IPv4 address

A Rendezvous Server MAY add a FROM parameter containing the original source IP address of a HIP packet (I1, R1, I2 or R2) whose source IP address has been rewritten. If one or more FROM parameters are already present, the new FROM parameter MUST be appended after the existing ones. Each time an RVS inserts a FROM parameter, it MUST also insert additional parameters that will be used to validate this and the subsequent HIP packets. These parameters are:

o An ECHO_REQUEST, containing a chunk of opaque data allowing to validate, in a possible subsequent answer, a TO parameter which MUST be protected by an ECHO_RESPONSE containing the same opaque data.

o A valid RVA_HMAC, protecting the packet integrity.

When a HIP node validates a FROM parameter, it is removed from the packet and recorded for later use (i.e., for building the corresponding TO parameter to be piggy-backed onto a subsequent answer). The packet's source IP address is also replaced by the address included in the first occurrence of FROM parameter.

For each FROM parameter, a HIP node MAY add to its replies a TO parameter containing the IP address included in the FROM. These replies will be sent via the RVS, which MUST remove the outer TO parameter from the packet and replace its destination address with the address contained in the TO parameter before relaying it.

5.2.5 TO Parameter Format and Processing

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                            Address                            |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
Type         65102 (under signature) or 65302 (after signature)
Length       16
Address      An IPv6 address or an IPv4-in-IPv6 format IPv4 address
```

A HIP node MAY add one or more TO parameter containing the final destination IP address of a HIP packet (I1, R1, I2 or R2) whose destination IP address needs to be rewritten by an RVS. This is essentially equivalent to loose source-routing. If one or more TO parameters are already present, the new TO parameter MUST be appended after the existing ones. Each time a node inserts a TO parameter, it MUST also insert additional parameters that will be used by the RVS for validation. These parameters are:

o An ECHO_RESPONSE, containing a chunk of opaque data allowing the RVS to validate the address contained in the TO parameter.

o A valid RVA_HMAC, protecting the packet integrity.

When the RVS validates a TO parameter, SHALL remove it from the packet, and SHALL replace the packet destination IP address with the address included in the TO parameter. Packet length and checksum MUST then be recomputed accordingly.

For each FROM parameter, a HIP node MAY add to its replies a TO parameter containing the IP address included in the FROM. These replies will be sent via the RVS, which MUST remove the outer TO parameter from the packet and replace its destination address field with the address contained in the TO parameter before relaying it.

5.2.6 VIA_RVS Parameter Format and Processing

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                            Address                            |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                               .                               .
.                               .                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                            Address                            |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
Type           65500
Length         Variable
Address        An IPv6 address or an IPv4-in-IPv6 format IPv4 address
```

At some point a, HIP endpoint might be in position to begin to send HIP packets directly towards the remote HIP endpoint's IP address, without further assistance from one or more of its RVS(s). In that case, it MAY include in these packets a subset of the IP address(es) of its RVSs for debugging purposes.

Similarly, a RVS relaying an I1 to the Responder or an R1 to the Initiator MAY include in these packets its IP address for debugging as well.

When the IP address of a RVS need to be included in a packet, by either an end-node or the RVS itself, one of these two methods is used:

- o Add RVS IP address into an existing VIA_RVS parameter situated at the end of the HIP packet, while modifying accordingly the size of the parameter.

- o Append a newly created VIA_RVS parameter at the end of the HIP packet if it does not already contain a VIA_RVS parameter.

Note that the main goal of using the VIA_RVS parameter is to allow operators to diagnose possible issues encountered while establishing a HIP association via a RVS.

5.3 Use of Existing HIP Messages and Parameters

5.3.1 ECHO_REQUEST and ECHO_REPLY Parameters

A FROM parameter MAY be augmented by including an ECHO_REQUEST parameter to the carrying packet. The contents of the ECHO_REQUEST might then be echoed back in ECHO_RESPONSE.

A TO parameter SHOULD be augmented and authenticated by including an ECHO_REPLY parameter to the carrying packet. The contents of the ECHO_REPLY MUST be copied from a previously received ECHO_RESPONSE.

All the HIP packets requiring RVS relaying facility to carry an answer packet SHOULD be augmented by the RVS with an ECHO_REQUEST parameter.

A possible packet answered via the RVS, thus requiring relaying facility, SHOULD be authenticated by an ECHO_REPLY parameter. The contents of the ECHO_REPLY MUST be copied from a previously received ECHO_RESPONSE.

On the receiving side, when a HIP node validates an ECHO_REPLY located after the signatures, it MUST remove it from the packet and recompute packet length and checksum accordingly.

5.3.2 REA Parameter

A HIP node associated via an RVS MAY use a REA parameter to make its correspondent aware of its veritable current IP address. If used, the REA parameter MUST be used in conformance with the guidelines specified in [5].

6. Diagram Notation

Notation	Significance
I, R	I and R are the respective source and destination IP addresses of the IP header
HIT-I, HIT-R	HIT-I and HIT-R are respectively the Initiator and the Responder HIT of the packet
R	The RVS_CAPABLE Control Field is set into the Control Field of the HIP header
REA:I	A REA parameter containing the IP address i is present in the HIP header
FROM:I	A FROM parameter containing the IP address I is present in the HIP header
TO:I	A TO parameter containing the IP address I is present in the HIP header
VIA:RVS	A VIA_RVS parameter containing IP addresses RVS is present in the HIP header
REDIR:R	A REDIRECT parameter containing IP address R of Responder is present in the HIP header
EREQ	An ECHO_REQUEST parameter is present in the HIP header
EREP	An ECHO_REPLY parameter is present in the HIP header
RREQ	A RVA_REQUEST parameter is present in the HIP header
RREP	A RVA_REPLY parameter is present in the HIP header

7. Establishing Rendezvous Associations

A HIP node that wants to register its IP address with its RVS MAY simply establish a HIP association with it. It MUST then keep its IP address current with the server by sending UPDATE packets whenever its set of IP addresses changes.

However, for the sake of economizing RVS resources, which can

possibly be used by several thousands of different HIP nodes, we define a new sort of "soft state" HIP association called a Rendezvous Association (RVA). In order to maintain this RVA established, a HIP Association need not remain established.

A HIP node MAY establish an RVA with its RVS by establishing a HA while adding an RVA_REQUEST parameter in an I2, possibly preceded by an I1 containing the same RVA_REQUEST. The possibility offered to initiate the protocol in I1 allows a HIP node to query a RVS for the set of offered rendezvous service types before completing the establishment of the Rendezvous association (in case the desired service type isn't available on this RVS). A RVS MUST then reply with, respectively, an R2 possibly preceded by an R1, which will both have the RVS_CAPABLE control field set, and contain a RVA_REPLY parameter specifying the characteristics of the offered RVA (validity time, type, etc.). Then, the RVS and the HIP node MAY delete most of the HIP Association state, retaining only the Lifetime, Initiator's HIT and IP address(es), as well as HIP_lg and HIP_gl integrity keys.

When a HA is established via an RVS, the integrity of HIP packets flowing between a HIP node and its RVS is protected by an additional RVA_HMAC keyed with these keys.

```
                     I1(I, RVS, HIT-I,
                        HIT-RVS)          +------+
            +--------------------------->|      |
            |+--------------------------|      |
            ||      R1(RVS, I, HIT-RVS,  |      |
            ||         HIT-I, R)         |      |
            ||                           | RVS1 |
            ||      I2(I, RVS, HIT-I,    |      |
            ||         HIT-RVS, RREQ)    |      |
            || +----------------------->|      |
            || |+-----------------------|      |
            || ||  R2(RVS, I, HIT-RVS,  +------+
            || ||      HIT-I, R, RREP)
            |V |V
           +-----+
           |     |
           |  I  |
           |     |
           +-----+
```

Figure 12: Establishing a Rendezvous Association

There is nothing to prevent an RVS node to advertise its RVS capabilities to the peers it associates with, nor to establish an RVA with another RVS.

If a HIP node wants to associate with several cascaded Rendezvous Servers RVS_i (0 < i < n+1), it SHALL sequentially create RVAs (RVA_i) with each of them, starting from the "nearest" (RVS_1) to the "farthest" (RVS_n). Apart from RVA_1, a node SHOULD create any such RVA_i (1 < i < n+1) by sending an I1 to RVS_i via each of the RVS which precede it, i.e., RVS_j (1 < j < i).

This is achieved by using (i - 1) different TO parameters containing, in order, the IP address of each RVS preceding RVS_i, i.e., RVS_j (1 < j < i). This process is similar to IP loose source-routing. Hence, A RVS accepting to be part of a cascade MAY relay an incoming I1 from one its clients to any given address and HIT. Those I1s MUST be protected by a valid RVA_HMAC parameter.

```
         I1(I, RVS1, HIT-I,                  I1(RVS1, RVS2, HIT-I,
         HIT-RVS2, TO:RVS2)      +------+       HIT-RVS2, EREQ1)
     +-------------------------->|      |----------------------------+
     |+--------------------------|      |<--------------------------+|
     || R1(RVS1, I, HIT-RVS2,    |      |   R1(RVS2, RVS1,          ||
     ||    HIT-I, R, EREQ1)      |      |      HIT-RVS2, HIT-I,     ||
     ||                          | RVS1 |      R, EREP1)            ||
     ||   I2(I, RVS1, HIT-I,     |      |                           ||
     ||      HIT-RVS2, RREQ,     |      | I2(RVS1, RVS2, HIT-I,     ||
     ||      EREP1, TO:RVS2)     |      |    HIT-RVS2, RREQ, EREQ1) ||
     || +----------------------->|      |------------------------+  ||
     || |+-----------------------|      |<----------------------+|  ||
     || || R2(RVS1, I, HIT-RVS2, +------+  R2(RVS2, RVS1,       ||  ||
     || ||    HIT-I, R, RREP,                 HIT-RVS2, HIT-I,  ||  ||
     || ||    EREQ1)                          R, RREP, EREP1)   ||  ||
     |V |V                                                      |V  |V
    +-----+                                                    +------+
    |     |                                                    |      |
    |  I  |                                                    | RVS2 |
    |     |                                                    |      |
    +-----+                                                    +------+
```

Figure 13: Establishing Cascaded Rendezvous Associations

8. Establishing HIP Associations via Rendezvous Servers

8.1 Sending a Redirect in Reply to I1

Instead of having the RVS relay incoming I1s to the correct Responder, one possibility is to answer with a REDIRECT packet when a HIP packet destined for one of the Rendezvous Server's HIP nodes arrives. This REDIRECT packet would contains the IP address and packet signature of the Responder.

The Responder cannot sign the redirect packets delivered by the RVS in real time. When the RVA is set up, the Responder sends the signed REDIRECT packet to the RVS, who stores it until the RVA expires.

By signing this REDIRECT packet and sending it to the RVS, the Responder is authorizing the Rendezvous Server's IP address to redirect Initiators to the Responder's IP address. The authorization is weak because the subject of the authorization is the IP address which is not bound to the HI of the Responder (similarly to what is described in , the possibility to use CGAs as IP addresses for RVSs might improve authorization security because the RVS might then prove to Initiators ownership of the CGA IP address, and the authorization issued to it to redirect to the Responder's IP address.

An implementation of this redirect packet is a R1 packet signed by the Responder, which contains an additional REDIRECT parameter (with the IP address of the Responder, and perhaps a limitation of the REDIRECT validity, like 'not-before' and 'not-after' dates, or hash chains) The RVS redirect an Initiator by replying to an I1 with this REDIRECT R1 in which the receiver HIT field has been field with the HIT of the Initiator. Note that this may expose the Initiator to replay attacks, but this is not very different from the situation where the Initiator receives a signed R1 whose signature also omits Receiver HIT.

```
                                        _____OFFLINE______
                                        R1(R, RVS, HIT-R
 I1(I, RVS, HIT-I, HIT-R) +----------+     HIT-0, REDIR:R)
 +------------------------|          |
 |                        |   RVS    |<-+-+-+-+-+-+-+-+-+-+
 |  +---------------------|          |                    |
 |  | R1(RVS, I, HIT-R,   +----------+                    +
 |  V    HIT-I, REDIR:RVS->R)                             |
+-----+            I1(I, R, HIT-I, HIT-R)           +-----+
|     |------------------------------------------->|     |
|     |<--------------------------------------------|     |
|  I  |            R1(R, I, HIT-R, HIT-I)           |  R  |
|     |            I2(I, R, HIT-I, HIT-R)           |     |
|     |------------------------------------------->|     |
|     |<--------------------------------------------|     |
+-----+            R2(R, I, HIT-R, HIT-I)           +-----+
```

Figure 14: Initiator redirected by Rendezvous Server with a Responder-signed R1

8.2 Passing I1 onto an ESP SA

If a HIP node and one of its Rendezvous Servers maintain a HIP Association, the Rendezvous Server MAY tunnel I1s incoming to this node's HIT into the corresponding ESP SA. The main drawbacks of this approach are that, (1) middle-boxes cannot see the encrypted I1 passing from an RVS to its clients, and (2) the source IP address of I1 is lost. In particular, (2) implies that the RVS MUST transmit to the Responder the original source IP address by either of the following:

- o add a FROM parameter to the HIP header
- o include the whole original IP header in the ESP payload (very similar to ESP tunnel mode)

o route back the subsequent R1 via the RVS

```
                                    ESP(RVS, R,
                                        I1(I, RVS, HIT-I,
    I1(I, RVS, HIT-I, HIT-R) +---------+      HIT-R, FROM:I))
    +----------------------->|         |--------------------+
    |                        |   RVS   |                    |
    |                        |         |                    |
    |                        +---------+                    |
    |                                                       V
   +-----+    R1(R, I, HIT-R, HIT-I, REA:R, VIA:RVS)    +-----+
   |     |<---------------------------------------------|     |
   |     |                                              |     |
   |  I  |            I2(I, R, HIT-I, HIT-R)            |  R  |
   |     |--------------------------------------------->|     |
   |     |<---------------------------------------------|     |
   +-----+             R2(R, I, HIT-R, HIT-I)           +-----+
```

Figure 15: Rendezvous Server Forwarding I1 onto an ESP SA

8.3 Rewriting I1 Destination IP Address

When a HIP packet destined for one of its HIP nodes arrives at a Rendezvous Server, it relays the packet to one of the HIP node's current IP addresses. In most case, it is expected that only the first packet of a HIP Base Exchange (i.e., I1) will require such relaying [2]. Subsequent packet of the HIP Base Exchange and all further data packets will directly flow between the HIP nodes, bypassing the Rendezvous Server. The RVA established between such a RVS and its peer has type I1_REWRITE_DST.

In the simplest case, the Rendezvous Server can relay an I1 towards its true destination by merely replacing the destination IP address of the I1 by one of the destination HIT owner's IP address(es). Note, however, that such I1s might be subject to egress filtering on the Rendezvous Server's network [9], thus causing I1 packet to be dropped (source IP address does not belong to the RVS network).

```
                                         I1(I, R, HIT-I,
    I1(I, RVS, HIT-I, HIT-R) +---------+    HIT-R, FROM:I)
    +------------------------>|         |--------------------+
    |                         |   RVS   |                    |
    |                         |         |                    |
    |                         +---------+                    |
    |                                                        V
   +-----+    R1(R, I, HIT-R, HIT-I, REA:R, VIA:RVS)    +-----+
   |     |<--------------------------------------------|     |
   |     |                                              |     |
   |  I  |            I2(I, R, HIT-I, HIT-R)            |  R  |
   |     |-------------------------------------------->|     |
   |     |<--------------------------------------------|     |
   +-----+             R2(R, I, HIT-R, HIT-I)           +-----+
```

Figure 16: Rendezvous Server Rewriting I1 Destination IP Address

8.4 Rewriting I1 Source and Destination IP Addresses

Because of egress filtering, a HIP Rendezvous Server might need to replace the original source IP address of an I1 by its own IP address, thus concealing the Initiator's IP address to the Responder.

While this might be desirable, one of the extension described in this document allows a Rendezvous Server to piggy-back incoming HIP packets with an OPTIONAL FROM parameter containing the original source IP address of the packet. A HIP node receiving a packet containing such a FROM parameter has two possibilities for answering back. It might answer an R1 back either:

- Directly to the IP address included in the FROM parameter. The RVA established between such a RVS and its peer has type I1_REWRITE_SRCDST.

- Via the Rendezvous Server IP address, adding to the R1 HIP header a TO parameter containing the IP address included in the FROM parameter. The RVA established between such a RVS and its peer has type I1R1_REWRITE_SRCDST.

```
                                        I1(I, RVS, HIT-I,
       I1(I, RVS, HIT-I, HIT-R) +---------+     HIT-R, FROM:I, VIA:RVS)
       +----------------------->|         |--------------------+
       |                        |   RVS   |                    |
       |                        |         |                    |
       |                        +---------+                    |
       |                                                       V
      +-----+     R1(R, I, HIT-R, HIT-I, REA:R, VIA:RVS)   +-----+
      |     |<-----------------------------------------------|     |
      |     |                                              |     |
      |  I  |            I2(I, R, HIT-I, HIT-R)            |  R  |
      |     |----------------------------------------------->|     |
      |     |<-----------------------------------------------|     |
      +-----+             R2(R, I, HIT-R, HIT-I)           +-----+
```

Figure 17: I1_REWRITE_SRCDST: Rendezvous Server Rewriting I1 Source and Destination IP Addresses

8.5 Rewriting I1 and R1 Source and Destination IP Addresses

It might be useful to relay further HIP packets (i.e., R1) via the RVS. For example, if the Initiator does not know the Responder's HIT, it will initiate an opportunistic exchange with the Responder via a RVS. The first problem is for the RVS to forward an I1 which doesn't have a destination HIT to the correct Responder.

Because an opportunistic Initiator uses the unspecified IPv6 address (i.e., ::0) as a place-holder for the Responder HIT in I1s it sends, an RVS cannot use this Responder HIT to demultiplex incoming "opportunistic" I1s. The only way to properly relay such Opportunistic I1s is for the RVS to lease per-HIT IP addresses, so the destination IP addresses of Opportunistic I1s can be used as a key to find the correct Responder.

In order to avoid trivial spoofing attacks with R1s, a HIP node receiving an opportunistic I1 from a Rendezvous Server MUST reply with its R1 via the same Rendezvous Server. Accordingly, an Initiator who has attempted an opportunistic exchange towards an IP address (those of the RVS) MUST discards all R1s received in answers which do not come from the same IP address. When sending the R1 via the RVS, the Responder MUST initiate the readdressing protocol as described in [5].

This restriction is made for security reasons. If the Initiator receives an R1 directly from the Responder, the only way to find the appropriate HIP state is to use as a key the RVS's IP address, which is possibly included in a VIA_RVS parameter. This solution MUST be

avoided because the VIA_RVS parameter is not trusted (The Initiator doesn't have a priori knowledge of the public key, and the included RVS IP address hasn't been "validated" by having the routing fabric delivers the IP header with this address as source). If this restriction is not made, a passive attacker might easily hijack a HIP state in I1_SENT state: it would learn a (source,destination) tuple of IP addresses in a flowing I1, then send to the source address a self-made R1 with a VIA_RVS parameter containing the destination address; that's it, the attacker hijacked the I1_SENT state. This an opportunity for eavesdropping, MitM, as well as DoS attacks.

Because these R1 packets are larger than I1 (they contain public keys and signatures), the relaying of such packet create an opportunity for denial of service attacks. To defend against these attacks, the Rendezvous Server needs to differentiate between legitimate HIP packets (i.e., I1 and subsequent HIP packets triggered by an I1) and illegitimate ones.

For the sake of reducing the load incurred on the RVS, an RVS is not required to keep track of IP addresses and other pieces of state associated with ongoing HIP exchanges. Such behavior is OPTIONAL. Instead, the relaying facility MAY make use of ECHO_REQUEST and ECHO_RESPONSE parameters.

Each time a packet is being relayed, the RVS MAY augment it with an ECHO_REQUEST parameter containing a chunk of opaque data. The receiver of such a packet SHOULD augment any packet answering to this packet with an ECHO_REPLY parameter containing the same chunk of opaque data. This opaque data allows an RVS to find and validate the answered packet IP addresses and HITs. When successfully validated, ECHO_REPLY parameters SHOULD be removed from the packet before relaying.

```
       I1(I, RVS,                   I1(RVS, R, HIT-I, HIT-0
          HIT-I, HIT-0)   +---------+   FROM:I, EREQ)
    +-------------------->|         |----------------------+
    |+--------------------|         |<--------------------+|
    || R1(RVS, I, HIT-R,  |   RVS   | R1(R, RVS, HIT-R,   ||
    ||    HIT-I, REA:R,   |         |    HIT-I, REA:R,    ||
    ||    VIA:RVS)        |         |    TO:I, EREP)      ||
    ||                    |         |                     ||
    ||                    +---------+                     ||
    |V                                                    |V
   +-----+             I2(R, I, HIT-R, HIT-I)          +-----+
   |     |------------------------------------------->|     |
   |     |<-------------------------------------------|     |
   |     |             R2(I, R, HIT-I, HIT-R)          |     |
   |  I  |                                             |  R  |
   |     |              ESP(R, I, SPI-R)               |     |
   |     |<-------------------------------------------|     |
   |     |------------------------------------------->|     |
   +-----+              ESP(I, R, SPI-I)               +-----+
```

Figure 18: I1R1_REWRITE_SRCDST: Responder replying via the RVS to an Opportunistic Initiator

8.6 Cascading Rendezvous Servers

In some situations, it might be useful to use cascaded Rendezvous Servers to establish RVS associations. A typical scenario would be a small number of "trusted" Rendezvous Servers and a larger number of "untrusted" Rendezvous Servers. Only the trusted Rendezvous Servers are aware of the IP addresses of the Responders. The untrusted servers know only the IP addresses of other (un)trusted Rendezvous Servers. Untrusted Rendezvous Servers are changed periodically, in order to lower the opportunity for flooding-type attacks on their IP addresses.

In the case of cascaded Rendezvous Servers, the parameters added to the HIP base exchange, like FROM, TO, VIA_RVS, ECHO_REQUEST/REPLY or RVA_HMAC, MUST be "aggregated" or "clustered" on a per-type basis. This means that, when an RVS needs to add onto a HIP packet a parameter which is already present in it, this parameter MUST be added just after the existing parameter(s) of the same type. For instance, a FROM parameter MUST be added just after the existing FROM(s) parameter(s). The same applies to TO, VIA_RVS, ECHO_REQUEST/REPLY or RVA_HMAC.

Another solution to cascaded Rendezvous Servers may be to encapsulate

the original packet into a PAYLOAD and then piggy-back it with additional parameters. This scheme has not been evaluated further.

```
                                                 I1(RVS2, R, HIT-I,
     I1(I, RVS,         I1(RVS1, RVS2,              HIT-R, EREQ1,
        HIT-I,             HIT-I, HIT-R,            EREQ2, FROM:I,
        HIT-R) +------+    EREQ1, FROM:I)  +------+ FROM:RVS1)
    +--------->|      |------------------->|      |---------+
    |          | RVS1 |                    | RVS2 |         |
    | +--------|      |<-------------------|      |<------+ |
    | |        +------+  R1(RVS2, RVS2,    +------+       | |
    | |                     HIT-I, HIT-R,                 | |
    | |                     EREP1, EREQ2,                 | |
    | |                     TO:I)                         | |
    | | R1(RVS1, I, HIT-R,             R1(R, RVS2, HIT-R, | |
    | |    HIT-I, REA:R,                  HIT-I, REA:R,   | |
    | |    EREQ2, EREQ1)                  EREP1, EREP2,   | |
    | |                                   TO:I, TO:RVS2)  | |
    | V                                                   | V
   +-----+    I2(I, R, HIT-I, HIT-R, EREP2, EREP1)     +-----+
   |     |-------------------------------------------->|     |
   |  I  |<--------------------------------------------|  R  |
   +-----+           R2(R, I, HIT-R, HIT-I)            +-----+
```

Figure 19: Two Cascaded Rendezvous Servers Relaying an I1-R1 Message Pair

8.7 Implication on the HIP integrity checks

The establishment of HIP associations via one or more Rendezvous Servers causes HIP packets flowing between the HIP nodes to be modified during transmission. Several kinds of modifications to both the IP and HIP headers are possible. The HIP protocol uses two kinds of packet integrity checks: hop-by-hop and end-to-end. The HIP checksum is a hop-by-hop check and SHOULD be verified and recomputed by each of the on-path HIP middle-boxes (e.g., Rendezvous Servers). The HMAC and SIGNATURE are end-to-end checks and MUST be computed by the sender and verified by the receiver.

8.7.1 Checksum

The checksum field of a HIP header to be modified MUST be verified before applying the modification and recomputed accordingly after.

8.7.2 HMAC and SIGNATURE

The HMAC and SIGNATURE field of a HIP header MUST be computed and

verified based on a "sender view" or "receiver view" of the HIP header. In particular, this implies that SIGNATURE and HMAC MUST NOT cover FROM and TO parameters added or removed by Rendezvous Servers and that the HIP pseudo-header used to compute and verify them MUST contain the IP addresses as seen by the remote HIP peer. In case of IP address concealment by the RVS, this means that the IP address of this RVS MUST be used in the pseudo-header in place of the IP address of the end host it conceals.

8.7.3 Example

Here is an example showing how to compute the different integrity checks (end-to-end and hop-by-hop) when two Rendezvous Servers are cascaded and conceals the Responder IP address (packet flowing along the path I -> RVS1 -> RVS2 -> R)

End-to-end integrity checks: HMAC and SIGNATURE are computed with a pseudo-header containing RVS1 as a place holder for the destination IP address, the rationale being that RVS1 is concealing the Responder IP address. Therefore, R will verify the signature using RVS1 as the destination IP address in the pseudo-header.

Hop-by-hop integrity checks: Checksum is computed hop-by-hop; first with I and RVS1, then with RVS1 and RVS2, and finally with RVS2 and R.

9. Security Considerations

The security aspects of different HIP rendezvous mechanisms are currently being investigated. This section describes the known threats introduced by these HIP extensions, and implications on the overall security of HIP and IP. In particular, the following tries to show that the extensions described in this document do not introduce additional threats in the Internet infrastructure.

It is difficult to encompass the whole scope of threats introduced by Rendezvous Servers because their presence have implications both at the IP and HIP layer. In particular, the extensions hereby described might allow for redirection, amplification and reflection attacks at the IP layer, as well as attacks on the HIP layer itself, for example Man-in-the-Middle attacks against the cryptographic core-protocol SIGMA used by HIP.

If an Initiator has an a priori knowledge of the Responder's HI when it first contacts it via the RVS, it has a means to verify the signatures in the HIP exchange, thus conforming to the SIGMA protocol which is resilient to Man-in-the-Middle attacks.

If an Initiator has not an a priori knowledge of the Responder's HI (so called Opportunistic Initiators), it is almost impossible to defend the HIP exchange against MitM attacks (cannot authenticate public keys exchanged). The only solution is to mitigate hijacking threats on the HIP state by requiring an R1 answering an Opportunistic I1 to come from the IP address where the I1 was initially sent. That way we retain a level of security which is equivalent to what exists today in the Internet: By sending an IP packet to an IP address, and receiving an answered IP packet from this same IP address, I know that the routing fabric trusts my correspondent to be represented by this IP address. While it is true that such security is weak, it is better than none, and avoids to introduce additional threats at the IP layer.

10. IANA Considerations

IANA needs to open a new registry for the Rendezvous Association (RVA) type. Defined RVA types are:

Type number	RVA Type
0	Reserved by IANA
1	I1_REWRITE_DST
2	I1_REWRITE_SRCDST
3	I1R1_REWRITE_SRCDST
4	I1_RELAY_ESP
5	I1R1_RELAY_ESP
6	REDIRECT
6-200	Reserved by IANA
201-255	Reserved by IANA for private use

Adding new reservations requires IETF consensus RFC2434 [14].

11. Acknowledgments

The following people have provided thoughtful and helpful discussions and/or suggestions that have improved this document: Marcus Brunner, Tom Henderson, Miika Komu, Mika Kousa, Pekka Nikander, Simon Schuetz,

Tim Shepard, Kristian Slavov, Martin Stiemerling, and Juergen Quittek.

Part of this work is a product of the Ambient Networks project, partially supported by the European Commission under its Sixth Framework Programme. It is provided "as is" and without any express or implied warranties, including, without limitation, the implied warranties of fitness for a particular purpose. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Ambient Networks project or the European Commission.

12. References

12.1 Normative References

[1] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[2] Moskowitz, R. and P. Nikander, "Host Identity Protocol Architecture", draft-ietf-hip-arch-00 (work in progress), October 2004.

[3] Nikander, P. and J. Laganier, "Host Identity Protocol (HIP) Domain Name System (DNS) Extensions", draft-ietf-hip-rvs-00 (work in progress), October 2004.

[4] Moskowitz, R., Nikander, P. and P. Jokela, "Host Identity Protocol", draft-ietf-hip-base-01 (work in progress), October 2004.

[5] Nikander, P., "End-Host Mobility and Multi-Homing with Host Identity Protocol", draft-ietf-hip-mm-00 (work in progress), October 2004.

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

[7] Vixie, P., Thomson, S., Rekhter, Y. and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, April 1997.

[8] Wellington, B., "Secure Domain Name System (DNS) Dynamic Update", RFC 3007, November 2000.

[9] Killalea, T., "Recommended Internet Service Provider Security Services and Procedures", BCP 46, RFC 3013, November 2000.

[10] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, May 2000.

12.2 Informative References

[11] Saltzer, J., "On the Naming and Binding of Network Destinations", RFC 1498, August 1993.

[12] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[13] Klensin, J., "Role of the Domain Name System (DNS)", RFC 3467, February 2003.

[14] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[15] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, July 2003.

Authors' Addresses

Julien Laganier
Sun Labs (Sun Microsystems) & LIP (CNRS/INRIA/ENSL/UCBL)
180, Avenue de l'Europe
Saint Ismier CEDEX 38334
FR

Phone: +33 476 188 815
EMail: ju@sun.com
URI: http://research.sun.com/

Lars Eggert
NEC Network Laboratories
Kurfuersten-Anlage 36
Heidelberg 69115
DE

Phone: +49 6221 90511 43
Fax: +49 6221 90511 55
EMail: lars.eggert@netlab.nec.de
URI: http://www.netlab.nec.de/

Appendix A. Document Revision History

Revision	Comments
00	Compared to draft-eggert-hip-rvs-00: Add 'Terminology' section. Remove sections about privacy (goes into the HIP RG RVS draft). Wrote 'Security Considerations' and 'IANA Considerations' sections. Add I1/R1 relaying to support Opportunistic Initiators. Complete REDIRECT packet description. Compared to draft-eggert-hip-rendezvous-00: Minor fixes to figures and their descriptive text. Added RVS protocol specification. Removed sections related to communications between HIP and non-HIP nodes. Use boilerplate from RFC 3668.

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.