HIP Research Group
Internet-Draft
Intended status: Informational
Expires: September 7, 2011

Z. Cao
H. Deng
F. Cao
China Mobile
March 6, 2011

HIP Extension for Flow Mobility Management
draft-cao-hiprg-flow-mobility-00

Abstract

This document defines flow mobility extension to the Host Identity Protocol (HIP). A multi-homed HIP host makes the binding of a flow and one or more locators, through the new parameter "E-LOCATOR", which is the extension of "LOCATOR" defined in RFC5206, the host can acknowledgement his peers with addresses available that fit for some traffic flow. Peer hosts then selects the most appropriate address to transfer the traffic flow.

Table of Contents

1. Introduction

The Host Identity Protocol (HIP) [RFC4423] uses a new identity, named host identities, instead of IP addresses, as host identities. Packets between two HIP hosts are forwarded by IP addresses but identified by the host identities. Thereby when the IP address of a host is changed, the connections between the host and its peers can be sustained. [RFC5206] encompasses messaging and elements of procedure for basic network-level mobility and simple multihoming. A general "LOCATOR" parameter for HIP messages that allows for a HIP host to notify peers about alternate addresses at which it is reachable is defined. The LOCATORs may be merely IP addresses, or they may have additional multiplexing and demultiplexing context to aid the packet handling in the lower layers.

To enable the traffic control, HIP could be extended to support the flow mobility. This document extends LOCATOR to support end-to-end flow mobility of HIP. The extended LOCATOR, named as E-LOCATOR, includes locators defined in RFC5206 and a flow identifier mobility option, which defines the flow that is suitable transferred through the corresponding locator. The detail format of E-LOCATOR is described in Section 4.

The motivations to do HIP flow mobility include:

- Enable the flow mobility in HIP. That means flow can be transferred through the most appropriate interface or redirected to a better interface or address according to some factors, such as address enable situations, user preference and operator policy, etc.

- Enable the load sharing. The traffic to a certain interface of host can be distributed among different interfaces. When the resource of one connection is limited, other interfaces can be used to help deliver the data together.

A flow is defined as a set of IP packets matching a traffic selector. A traffic selector can identify the source and destination IP addresses, transport protocol number, the source and destination port numbers and other fields in IP and higher layer headers. For more flow information, please refer to [RFC6089].

2. Scenarios

End-to-end flow mobility is important to HIP multihoming. The traffic control, charging, QoS control and other operations can be operated based on flow.

A host that has one interface with multiple addresses, or a host that has multiple interfaces, each interface has a separate address are both multi-homed HIP hosts. It is envisioned that a multi-homed host can use several addresses simultaneously to transfer flows.

When different addresses are used simultaneously to transfer flows, first there must be policies in the multi-homed host about deciding a flow to be transferred through a certain address; we call this as flow binding. Then end-to-end address chosen and readdress are both necessary. Before a communication, the multi-homed host should be able to inform its peer about the reachable addresses, with the corresponding flow binding; peers should be able to choose the most suitable address for communication according to the flow going to be transferred; during the conversation, caused by IP address changing or in order to realize load balance, due to some mechanism, the multi-homed host may redirect some exiting flows with its peer from a previous interface or address to a new interface or address.

These situations are typical flow mobility scenarios. In these scenarios, there is a need for some helper functionality in the network, such as a HIP rendezvous server [RFC5204]. Such functionality is out of the scope of this document.

3. Protocol Operations

This protocol is based on "End-Host Mobility and Multihoming with the Host Identity Protocol" [RFC5206] .

This section introduces the solution of flow mobility. Using the parameters "E-LOCATOR" introduced in this specification, a multi-homed HIP host can notify peers about alternate addresses with corresponding flow mobility option; a flow can be identified by a FID in the mobility option. We can assume this as flow binding. Then the peers can select the most suitable address as the communication address. During the communication, when the using address is changed or in order to make load balance, the multi-homed host can redirect the existing flows to other addresses by using E-LOCATOR.

3.1. Flow binding

It is assumed that there should be some policies of flow binding. A flow binding in the multi-homed host is about a flow to be transferred through a certain address. In E-LOCATOR, a locator is followed by a "Flow Identification Mobility Option", which means flow with FID in the option is going to be transferred through the locator. The details of these policies are outside the scope of this document.

In this document, a host with HIP protocol that initializes a connection is Initiator; its peer host is Responder[RFC4423].

3.2. Base Exchange

Assuming that the Responder host has multiple addresses available at begin of the communication with its peer. When Initiator initializes the Base Exchange, a Responder host may include an E-LOCATOR parameter in the R1 packet that it sends to the Initiator. This parameter MUST be protected by the R1 signature.

The procedure of Base Exchange with RVS is followed:

1. First of all, Responder registers a RVS service with a RVS server; its current available IP addresses are maintained by the RVS.
2. An Initiator initializes the Base Exchange. First, it sends I1 packet with Initiator's and may be Responder's HIT, to the RVS, with which the Responder registers. The source IP address of I1 is Initiator's IP address. The destination IP address of I1 is RVS's IP address that can be got from a DNS server or other servers.
3. Then the RVS found that I1 is aimed to the Responder, so it updates the head of I1 packet and forwards it to the Responder. The source IP address of I1 is RVS's IP address. The destination address is currently available IP address of the Responder [RFC5204].
4. After authentication, the Responder sends R1 packet to the Initiator; an E-LOCATOR parameter is included in R1. This parameter is protected by the R1 signature. Currently available IP addresses of the Responder with corresponding flow are list in E-LOCATOR.
5. When the Initiator gets the R1 packet, according to the flow that to be transferred, it chooses the most suitable address among the entire addresses list in the E-LOCATOR, that is to say, choose the locator with the FID the same as the flow to be transferred. If there is only one locator in the parameter, then the Initiator chooses it as the communication address. If there is no locator with the corresponding flow, then the Initiator may choose the preferred locator to use. The Initiator should set the status as ACTIVE once an address has been determined and send the I2 packet to the new choose address. The I1 destination address and the new choose address may be identical. All new other locators must still undergo address verification once the Base Exchange completes [RFC5206].

During the Base Exchange, as the Initiator knows what kind of flow is to be transferred, it can make its most suitable address as the

source address. The Initiator may include one or more E-LOCATOR parameters in the I2 packet, independent of whether or not there was a E-LOCATOR parameter in the R1. These parameters must be protected by the I2 signature. Even if the I2 packet contains E-LOCATOR parameters, the Responder must still send the R2 packet to the source address of the I2. The new choosing address by the Responder should be identical to the I2 source address. If the I2 packet contains E-LOCATOR parameters, all new locators must undergo address verification as usual, and the ESP traffic that subsequently follows should use the addresses determined during the Base Exchange.

3.3. Flow Mobility

When a multi-homed host moves to a new place, the available address may be changed or there may be a new address available and the new address is more suitable for the existing flow, the multi-homed host can send UPDATE message to its peer to inform the new available or new more suitable address and then redirect the existing flow to the new address.

3.3.1. Readdress without Rekey

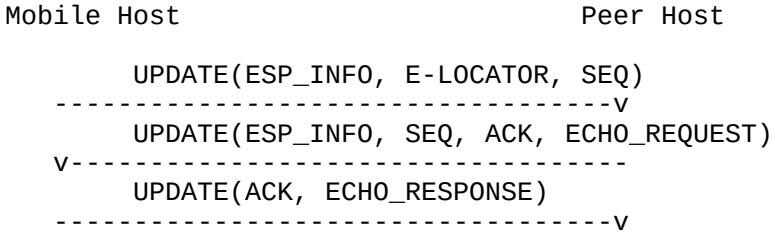


Figure 1: Readdress without Rekey

According to RFC5206, during the procedure of readdressing, hosts can use the old SAs or create new SAs. The first example considers the case in which no rekeying occurs on the SAs and the new IP address are within the same address family (Ipv4 or Ipv6) as the first address. The scenario is depicted in Figure 1.

1. The multi-homed host is disconnected from the peer host for a short period of time while it switches from one IP address to another. Upon obtaining a new IP address, the multi-homed host sends an E-LOCATOR parameter to the peer host in an UPDATE message. The same FID as existing flow must be included with the new locator. Set of ESP_INFO and SEQ parameters are the same as RFC5206 3.2.1 depicts.

2. When the peer host receives the UPDATE message, it performs as RFC5206 3.2.1 depicts.
3. The multi-homed host completes the readdress by processing the UPDATE ACK and echoing the nonce in an ECHO_RESPONSE. Once the peer host receives this ECHO_RESPONSE, it considers the new address to be verified and can put the address into full use.
4. The existing ESP traffic flow is transferred to the new address.

3.3.2. Readdress with Multi-homed-Initiated Rekey

If the Multi-homed host decide to rekey the SAs at the same time that it notifies the peer of the new address. In this case, the above procedure described in Figure 2 is slightly modified. The UPDATE message sent from the Multi-homed host includes an ESP_INFO with the OLD SPI set to the previous SPI, the NEW SPI set to the desired new SPI value for the incoming SA, and the KEYMAT Index desired. Optionally, the host may include a DIFFIE_HELLMAN parameter for a new Diffie- Hellman key. The peer completes the request for a rekey as is normally done for HIP rekeying, except that the new address is kept as UNVERIFIED until the UPDATE nonce challenge is received as described above. Figure 2 illustrates this scenario.

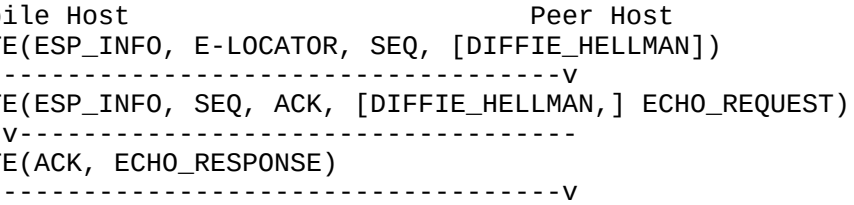


Figure 2: Readdress with Multi-homed-Initiated Rekey

3.3.3. Load balance

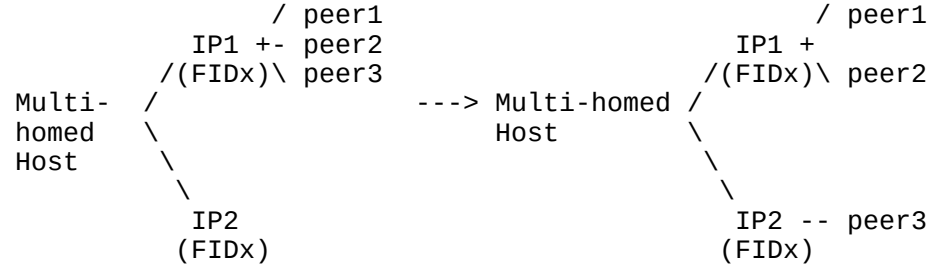


Figure 3: Load Balance

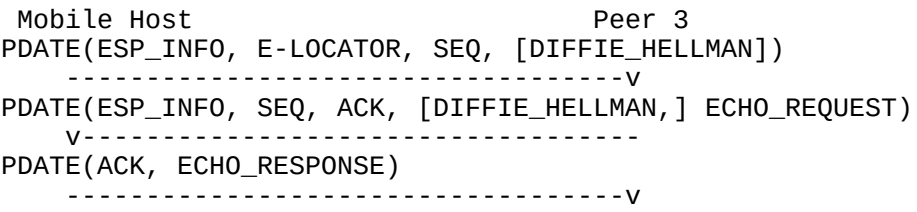


Figure 4: Flow Redirection

Considering the scenario that the multi-homed host has two address IP1, I23, which both are suitable for transferring flow X, identified by FIDx. Peer1 host and Peer2 both have flow X with multi-homed host through IP1. A new flow X is started between multi-homed host and Peer3, also using IP1. Then in order to make load balance, multi-homed host decides to redirect flow X with Peer3 to IP3. It then sends an UPDATE message to Peer 3. E-LOCAOR is included in the message, and there is only one locator, carries IP3 with FIDx option in the parameter. Once receiving the UPDATE message, since there is only one address available for FIDx, Peer3 redirects the flow X to IP3. The scenario is depicted as Figure 3. There must be policies for a multi-homed host to decide when to redirect the flow and which address is redirected to, the policies are out scope of this document.

4. E-Locator Definition

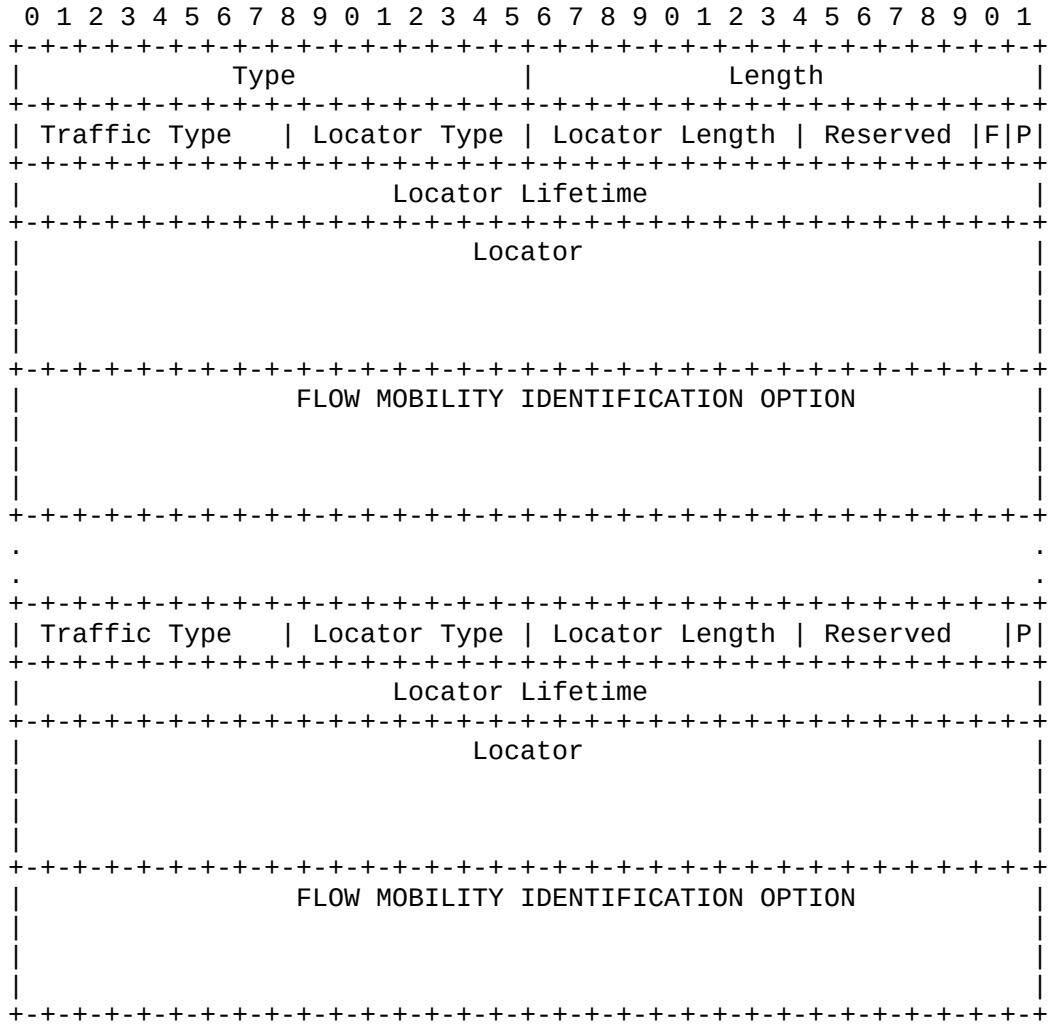


Figure 5: E-Locator

F Flag

A new flag, when the locator carries a corresponding flow identification mobility option, it is set to 1; otherwise it is set to 0, that means the locator is suitble for all flow;

Flow Identification Mobility Option: as defined in [RFC6089]

5. Security Considerations

TBD.

6. IANA Considerations

This document does not require any IANA actions.

7. Normative References

[RFC4423] Moskowitz, R. and P. Nikander, "Host Identity Protocol (HIP) Architecture", RFC 4423, May 2006.

[RFC5204] Laganier, J. and L. Eggert, "Host Identity Protocol (HIP) Rendezvous Extension", RFC 5204, April 2008.

[RFC5205] Nikander, P. and J. Laganier, "Host Identity Protocol (HIP) Domain Name System (DNS) Extensions", RFC 5205, April 2008.

[RFC5206] Nikander, P., Henderson, T., Vogt, C., and J. Arkko, "End-Host Mobility and Multihoming with the Host Identity Protocol", RFC 5206, April 2008.

[RFC6089] Tsirtsis, G., Soliman, H., Montavont, N., Giaretta, G., and K. Kuladinithi, "Flow Bindings in Mobile IPv6 and Network Mobility (NEMO) Basic Support", RFC 6089, January 2011.

Authors' Addresses

Zhen Cao
China Mobile
28 Xuanwumenxi Ave,Xuanwu District
Beijing 100053
China

Email: zehn.cao@gmail.com

Hui Deng
China Mobile
28 Xuanwumenxi Ave,Xuanwu District
Beijing 100053
China

Email: denghui@chinamobile.com

Feng Cao
China Mobile
28 Xuanwumenxi Ave,Xuanwu District
Beijing 100053
China

Email: fengcao@chinamobile.com

HIP Research Group
Internet-Draft
Intended status: Informational
Expires: September 7, 2011

Z. Cao
F. Cao
H. Deng
China Mobile

March 6, 2011

Communication between a HIP-enabled Host and a Legacy Host
draft-cao-hiprg-legacy-host-00

Abstract

This document describes a way to support a HIP-enabled host to have the ability to communicate with legacy hosts. By leveraging a proxy to bridge the communication between HIP host and non-HIP hosts, this document does not need the extensions to the HIP protocol.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 7, 2011.

Copyright Notice

Table of Contents

1. Introduction

HIP protocol [RFC5201] establishes an IP-layer communications context (called HIP association) between two hosts prior to communications. It also defines a packet format and procedures for updating an active HIP association. HIP offers many well-defined features to end hosts such as security, integrity protection, mobility and multi-homing.

HIP was designed to work with unmodified applications[RFC5338]. To ease incremental deployment, it is important to support the transition of legacy hosts towards HIP-enabled host. In the first phrase of this transition, some legacy hosts are turning on their abilities to support HIP, but most of them, especially internet servers have not been HIP-aware. There should be a mechanism that supports a HIP-enable host to communicate with non-HIP enabled legacy hosts.

[I-D.melen-hip-proxy] defines HIP protocol extensions that allow a non-HIP host to use the services of a HIP-aware proxy node and have capabilities to communicate with a HIP host.
[I-D.irtf-hiprg-proxies] investigates the HIP proxies for both directions. The scenario introduced in this document is slightly different in that the proxy is located in the local network when serving the HIP host to communicate with the non-HIP host.

This document describes a way to support a HIP-enabled host to have the ability to communicate with legacy hosts. By leveraging a proxy to bridge the communication between HIP host and non-HIP hosts, this document does not need the extensions to the HIP protocol.

2. Proposed Mechanism

2.1. Overview

In [I-D.melen-hip-proxy], HIP-proxy generates a Host Identity for each legacy host it will represent in the network. Instead of creating a static identity for each legacy host, this document specifies a mechanism of proxy dynamically assigning a host identity for each non-HIP host. The proxy will serve as the end point of the HIP base exchange. After the HIP association is established, the proxy will be in charge of translation between the HIP packet and legacy IP packets. The proxy will encapsulates the HIP packet payload into the IP header, so that the non-HIP host will participate into the communicate as usual. [I-D.irtf-hiprg-proxies]

This solution does not introduce any HIP protocol extention. The proxy that intercepts the communication will take the responsibility of holding the communication. Both the HIP aware initiator and non-HIP responder will be kept transparent about the proxy.

2.2. Proposal Walkaround

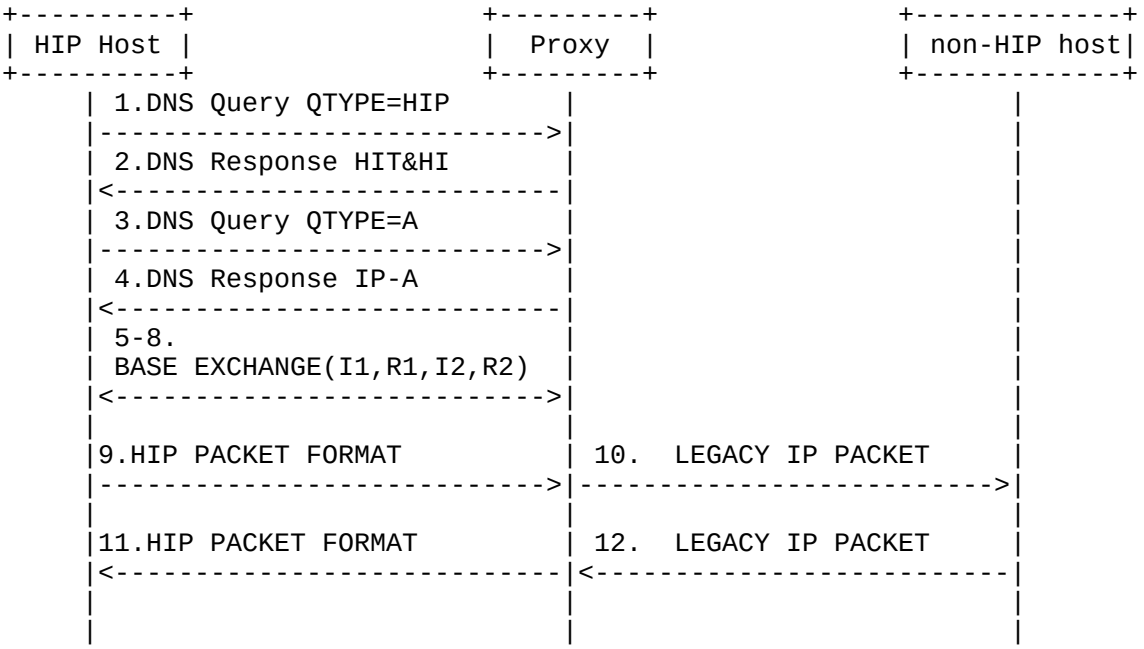


Figure 1: Proposed Mechanism

Most applications translate the domain name into one of more IP addresses before data plane communication. HIP is no exception. The HIP-enabled host first launches the DNS query to retrieve the remote host's HI/HIT or RVS address. Without knowing if the remote host supports HIP-based exchange, the HIP host is expecting to receiving the remote host HIP based Identities.

The proxy, which intercepts the DNS query, will iteratively forward the query to the global DNS to find an answer. If the responder is HIP enabled, it will have its HI or HIT registered in the DNS and the proxy will get an answer. However, if the responder is not HIP aware, and only have type A or AAAA records in the DNS system, the query for QTYPE=HIP will fail. On detecting that the responder is not HIP aware, the DNS proxy will use a temporary HI/HIT (T-ID) generated locally and reply this temporary HI/HIT to the initiator. The proxy will associate the T-ID with the IP address of the responder. After the HIP RR query reponse, the Type-A query response is followed, via which the initiator get the the IP address of the proxy node.

The HIP base exchange will proceed between the initiator and the proxy (step.5-8). Then, the HIP association is established between the initiator and the proxy, i.e., between the host's HI and the temporary HI assigned to the reponder by the proxy. If the initiator starts data communication towards the responder, the proxy on the data path will be responsible for the tranlation between HIP packets and IP packets. First, the proxy will de-capsulate the packet and decrypte the packet to get the original IP packet inside. By inspecting the HIP header after the IP header, the proxy is aware of the destination's HIT/LSI. If the HIT and LSI is mapped to one of the responder's IP address, the proxy will translate the packet with the destination address as the responder's IP address, and source address as the proxy IP address. The destination port is kept unchanged, but the source port can be dynamically assigned.

2.3. Host Identity Management

TBD.

3. Security Considerations

TBD.

4. IANA Considerations

This document does not require any IANA actions.

5. Normative References

[I-D.irtf-hiprg-proxies]
Zhang, D., Xu, X., and S. Shen, "Investigation in HIP Proxies", draft-irtf-hiprg-proxies-01 (work in progress), October 2010.

[I-D.melen-hip-proxy]
Melen, J., Ylitalo, J., and P. Salmela, "Host Identity Protocol-based Mobile Proxy", August 2009, <draft-melen-hip-proxy-02.txt (work in progress)>.

[RFC5201] Moskowitz, R., Nikander, P., Jokela, P., and T. Henderson, "Host Identity Protocol", RFC 5201, April 2008.

[RFC5205] Nikander, P. and J. Laganier, "Host Identity Protocol (HIP) Domain Name System (DNS) Extensions", RFC 5205, April 2008.

[RFC5338] Henderson, T., Nikander, P., and M. Komu, "Using the Host Identity Protocol with Legacy Applications", RFC 5338, September 2008.

Authors' Addresses

Zhen Cao
China Mobile
28 Xuanwumenxi Ave,Xuanwu District
Beijing 100053
China

Email: zehn.cao@gmail.com

Feng Cao
China Mobile
28 Xuanwumenxi Ave,Xuanwu District
Beijing 100053
China

Email: fengcao@chinamobile.com

Hui Deng
China Mobile
28 Xuanwumenxi Ave,Xuanwu District
Beijing 100053
China

Email: denghui@chinamobile.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 7, 2011

D. Zhang
X. Xu
Huawei Technologies Co.,Ltd
J. Yao
CNNIC

March 6, 2011

Investigation in HIP Proxies
draft-irtf-hiprg-proxies-02

Abstract

HIP proxies play an important role in the transition from the current Internet architecture to the HIP architecture. A core objective of a HIP proxy is to facilitate the communication between legacy (or Non-HIP) hosts and HIP hosts while not modifying the host protocol stacks. In this document, the legacy hosts served by proxies are referred to as Legacy Hosts (LHs). Currently, various design solutions of HIP proxies have been proposed. These solutions may be applicable in different working circumstances. In this document, these solutions are investigated in detail and compare their effectiveness in different scenarios.

Requirements Language

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 [RFC2119].

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 7, 2011.

Copyright Notice

Table of Contents

1. Introduction

The Host Identity Protocol (HIP) is a ID/Locator separating technology for Internet Protocol (IP) networks. It introduces a new host identifier layer in the middle of the network layer and the transport layer so as to comprehensively address the issues of mobility, multi- homing. Compared with other ID/Locator separating technologies, HIP is security integrited. The Host Identities (HIs) of HIP enable hosts are verifiable by using cryptographic methodology. Particularly, HIP provides a handshaking process for HIP hosts to authenticate each other and distribute symmetric keys for securing subsequent communication. Therefore, a HIP host cannot directly communicate with a legacy host.

As core components of HIP extensional solutions, HIP proxies have attracted increasing attention from both the industry and the academia. A HIP proxy is a middlebox located between a legacy host and a HIP enabled host for protocol transition. Under the assistance of a HIP proxy, a legacy host can communicate with its desired HIP host without updating its protocol stack.

Currently, multiple research work is engaged in both design and performance assessment of HIP proxies. In this document, we attempt to investigate several important designing solutions and compare their effectiveness in different scenarios. Actually, there has been a detailed discussion of HIP proxies in [SAL05]. This document can be regarded as a complement of that paper. Some new topics (e.g., the asymmetric path issues occurred in the load-balancing mechanisms for HIP proxies and the necessity of extending the HIP RR for HIP proxies) are discussed in the draft. Theoretically, LHs and the HIP hosts which the LHs intend to communicate with can be located anywhere in the network. However, in this document, without mentioning otherwise, it is assumed that legacy hosts are located within a private network and HIP hosts are located in the public network, since this is the most important scenario that HIP proxies are expected to support [SAL05].

The remainder of this document is organized as follows. Section 2 lists the key terminologies used in this document. In section 3, the essential functions of HIP proxies are indicated, and a classification of HIP proxies is proposed to benefit subsequent analysis. In section 4, we analyze the issues that HIP proxies have to face in constructing a Load Balancing Mechanism (LBM) which facilitates communication initiated by LHs. Section 5 analyzes the issues that HIP proxies in a LBM have to face if they also need to support communication initiated by HIP hosts. In section 6, we investigate the issues that HIP proxies have to deal with in supporting dynamic load balancing and redundancy. Section 7 provides

a brief discussion about the influence brought by DNSSEC [RFC4305] to the deployment of HIP proxies. Section 8 is the conclusion of the entire document. Section 9 is the security consideration.

2. Terminology

BEX: HIP Base Exchange

DI Proxy: DNS Inspecting Proxy

HA: HIP association

LBM: Load Balancing Mechanism

N-DI proxy: Non-DNS Inspecting Proxy

LHs: Legacy Hosts which are made up as HIP hosts by HIP proxies.

3. HIP Proxies

3.1. Essential Functionality of HIP Proxies

A primary function of HIP proxies is to facilitate the communication between legacy (or Non-HIP) hosts and HIP hosts while not modifying the host protocol stacks. In order to achieve this, a HIP proxy needs to intercept the packets transported between LHs and HIP hosts before they arrive at their destinations. Upon capturing such a packet, a HIP proxy needs to transfer it into the format which can be recognized by the host which the packet aimes to.

Assume a proxy captures a packet sent out by a LH. If the packet is destined to a HIP host, the proxy first tries to find out whether there is an appropriate HIP association (HA) in its local database to process the packet. If the HA exists, the proxy then uses the key maintained in the HA to encrypt the payload in the packet, transfer the packet into the HIP compatible fromat, and forwardes it to the HIP host. However, if there is not a proper HA, the proxy needs to use the HI and HIT assigned to the LH to carry out a HIP Base Exchange (BEX) and generate a new HA with the HIP host. The newly generated HA is then maintained in the local database.

Similarly, when processing a packet from a HIP host, the proxy needs to find a proper HA and use the keying material in the HA to decrypt the packet, and thus the packet is transferred into an ordinary IP packet and forwarded to the legacy host.

3.2. A Taxonomy of HIP Proxies

In practice, there are various design alternatives for HIP proxies. To benefit the analysis, in this document HIP proxies are classified into DNS lookups Intercepting Proxies (DI proxies) and Non-DNS lookups Intercepting Proxies (N-DI proxies). As indicated by the name, a DI proxy needs to intercept DNS lookups in order to correctly process the follow-up communication between LHs and HIP hosts, while N-DI proxies do not have to.

Note that a DI proxy implementation may also be able to intercept the lookup between a LH and a resolution server other than DNS. However, currently DNS is the only resolution mechanism detailed analyzed and extended to support HIP communication. Hence, DNS is only resolution mechanism considered in this document.

To avoid confusion, in the remainder of this document, the terms "lookup" and "answer"are used in specific ways. A lookup refers to the entire process of translating a domain name for a legacy host. The answer of a lookup is the response from a resolution server which terminates the lookup.

3.3. DI Proxies

Usually, before a legacy host communicates with a remote host, the legacy host needs to consult a DNS server for the IP address of its destination. On this premise, a DI proxy can effectively identify the HIP hosts which legacy hosts may contact in near future by intercepting DNS lookups.

In practice, it is common to deploy one or multiple DNS resolvers for a private network. A host in the private network can thus send its queries to a resolver instead of communicating with authoritative DNS servers directly. If the resolver does not cache the inquired RRs, it will try to collect them from authoritative DNS servers. In a lookup process, a resolver may have to contact multiple authoritative DNS servers before it eventually gets the desired DNS RRs.

On the occassions where a resolver is located out of a private network, a HIP proxy located at the border of the network can intercept the DNS queries from LHs and then use the FQDNs obtained from the queries to initiate a new DNS lookup to the resolver to inquire about the desired information (AAAA RRs, HIP RRs, and etc.). If the host that the legacy host intends to communicate with is HIP enabled, the DNS resolver will hand out a HIP RR associated with an AAAA RR to the proxy. After maintaining the needed information (e.g., HITs, HIs, and IPs addresses of the HIP hosts) in the local database for future usage, the proxy returns an answer with an AAAA

RR to the legacy host.

When the resolver is located inside the private network, conditions are a little more complex. If a proxy is deployed on the path between LHs and the resolver, it can operate as same as what is illustrated in the above paragraph. The proxies which can be deployed in this way are introduced in the remainder of this sub-section. However, if a proxy is located at the border of the network (i.e., in the middle of the resolvre and authoritative DNS servers), the proxy has to intercept the DNS lookups between the resolver and authoritative DNS servers. Because the resolver may have to contact multiple authoritative DNS servers to get a desired answer, for efficiency, the proxy can only inspect the responses from DNS services and find out DNS answers. Because the answer of a DNS lookup does not contain any NS RR, it can be easily distinguished from the intermediate responses. After identifying a DNS answer, a DI proxy can locate the DNS server maintaining the desired RRs from the packet header and identify the FQDN of the inquired host from the packet payload. Then, the proxy initiates an independent lookup to the DNS server to check whether the host is HIP enabled. If it is, the proxy maintains the information of the host for future usage and returns an answer with an AAAA RR to the resolver.

Besides intercepting DNS lookups, some kinds of DI proxies also modify the contents of the AAAA RRs in DNS answers to ehance the efficiency or deploying flexibility. For instance, [RFC5338] indicates that a HIP proxy can returns HITs rather than IP addresses in DNS answers to LHs. Consequently, the data packets from LHs to HIP hosts will use the HITs of the HIP hosts as destination addresses. The HIP proxy can then advertise a route of the HIT prefix to attract the packet for HIP hosts. [PAT07] also proposes a proxy solution which requires a HIP proxy to maintain an IP address pool. When sending a DNS answer to a LH, the proxy select an IP address from its pool and inserts it in the answer. The legacy host will regard this IP address as the IP address of the peer it intends to communicate with. In the subsequent communication, when the host sends a packet for the remote HIP host, it will use the IP address assigned by the proxy as the destination address. Therefore, the HIP proxy can intercept the packets for the HIP hosts by advertising the routes of the IP addresses in its pool. In the remainder of this document, these two types of proxies are referred to as DI-HIT proxies and DI-NAT proxies respectively, and the DI proxies which do not modify the contents of DNS answers (i.e., return the IP addresses of HIP hosts in answers) are referred to as DI-transparent proxies.

Compared with DI-HIT and DI-NAT proxies, DI-transparent proxies show their limitations in multiple aspects. For instance, it is a practical solution for a LH to publish the IP address of its proxy in

its DNS AAAA RR so that the packets for the host will be directly forwarded to the proxy. Therefore, when a LH served by a DI-transparent proxy attempts to communicate with two remote LHs served by a same HIP proxy, it is difficult for the host to distinguish one remote host from the other as they both use the same IP address. In addition, DI-transparent proxies cannot work properly in the circumstance where HIP hosts renumber their IP addresses during the communication due to, e.g., mobility or re-homing. For DI-HIT or DI-NAT proxies, these issues can be largely mitigated as the IP addresses of HIP hosts will never be used by DI-HIT or DI-NAT proxies to identity hosts.

Moreover, it is difficult for DI-transparent proxies to advertise routing information to attract the packets transported between LHs and remote HIP hosts. Consequently, they can be only deployed at the borders of private networks. DI-HIT (or DI-NAT) proxies, however, can easily attract the packets for HIP hosts to themselves because the packets destined to HIP hosts use HITs (or the IP addresses selected from pools) as their destination addresses. Hence, they can locate inside the networks. Therefore, in private networks which resolvers are located inside, DI-HIT or DI-NAT proxies can be deployed on the path between the resolvers and LHs and reduce the overhead on the proxies imposed by intercepting DNS lookups.

It is recommended to use DNSSEC [RFC4305] to prevent attackers from tampering or forging DNS lookups between resolvers and DNS server. This solution may affect the deployment of HIP proxies. For instance, DI-HIT and DI-NAT proxies need to modify the contents of DNS answers, and thus they should be only deployed on the path between legacy hosts and their resolvers if DNSSEC is deployed. Therefore, a DI-HIT proxy (or a DI-NAT proxy) cannot not be deployed in the middle of DNSSEC-enabled resolvers and authoritative DNS servers.

3.4. N-DI Proxies

Unlike DI proxies, an N-DI proxy does not try to intercept DNS lookups transported between LHs (or resolvers) and DNS servers.

In the circumstances where the HIP hosts that LHs intend to contact are predicable, an N-DI proxy can maintain a list of the information of the HIP hosts [SAL05]. After intercepting a packet from a LH, the proxy can ensure the packet is for a HIP host if the destination address of the packet is maintained in the list.

In the circumstances where it is difficult to predicate the HIP hosts that LHs intend to contact in advance, an N-DI proxy needs to contact DNS servers to find out whether the destination IP address of a

packet belongs to a HIP host or a legacy host. The information obtained from the DNS servers can be maintained within two lists. One list is for the information of HIP hosts; the other is for the information of legacy hosts. When intercepting a packet, the N-DI first compares the destination address of the packet against the addresses in the lists to find out whether the destination of the packet is HIP enabled. If the address is not maintained in the lists, the proxy then has to consult resolution systems and maintains the information of the host which the packet is aimed for in the correspondent list, according the answers from resolution systems.

3.5. Distributed Implementation of DI Proxies

As discussed above, DI proxies have to intercept the DNS lookup packets between legacy hosts and DNS servers in order to facilitate the communication between LHs and HIP hosts. This requires a DI proxy be deployed on the boundary of the private network or on the path where LHs and the DNS resolver tranport their lookup packets. In the circumstance where DNSSEC is deployed, a DI proxy cannot even be deployed on the boundary of the private network either, if the proxy needs to modify DNS lookup packets. Such inflexibility may be undesired under certain circumstances.

In this section, we analyze a design option of DI proxies which improves the deployment flexibility of DI proxies and avoids the issue brought by DNSSEC by separating the DNS related functionality (i.e., intercepting and modifying the DNS communication) from DI proxies. The component performing the DNS lookup interception is referred to as the DNS lookup inspector while the component performing the protocol transfermation is referred to as the proxy component. A DNS lookup inspector is located in a place where it can intercept and modify DNS lookups. In practice, a DNS resolver can also be extended to act as an inspector.

3.5.1. Distributed DI-HIT Proxies

The DNS lookup inspector of a distributed DI-HIT proxy returns HITs in DNS answers to LHs. Therefore, the associated DI-HIT proxy can advertise routing information inside the private network to attract the packets using HITs as destination addresses. Additionally, the inspector needs to transfer other information (e.g, IP addresses of the HIP hosts and RVSes) of the HIP hosts contained in DNS RRs to the DI-HIT proxy component so that the proxy can perform BEXes with the HIP hosts on behavior of LHs.

A DI-HIT proxy component can be associated with multiple DNS lookup inspectors. It is possible for a DI-HIT proxy component to be deployed in public networks to support multiple private networks.

This property is useful when Internet services providers (ISPs) intend to facilitate the legacy hosts in the private networks without HIP proxies to communicate with HIP hosts.

3.5.2. Distributed DI-NAT Proxies

A DNS lookup inspector of a distributed DI-NAT proxy needs to not only return the IP addresses in the address pool of the DI-NAT proxy component but also transfer the mapping information of the IP addresses and the correspondent HIP hosts to the DI-NAT proxy component. Moreover, the resolver needs to maintain the mapping information so as to assign an IP address for multiple HIP hosts concurrently.

Similar with Distributed DI-HIT Proxies, a DI-NAT proxy component can also be deployed in a public network. In this case, the addresses in the address pool must be routable in the public network.

3.5.3. Distributed DI-transparent Proxies

A DNS lookup inspector of a distributed DI-transparent proxy needs not to modify DNS answers, but it needs to transport the IP addresses and HIs of queried HIP hosts to the DI-NAT proxy component. In this case, a DI-transparent proxy component must be deployed on the boundary of the private network in order to guarantee that it can intercept packets.

4. Issues with LBMs in Supporting LHs to Initiate Communication

If there is only a single HIP proxy deployed for a private network, the proxy may become the cause of a single-point-of-failure. In addition, when the number of the users increases, the overhead imposed on the proxy may overwhelm its capability, which makes it the bottleneck of the whole mechanism. A typical solution to mitigate this issue is to organize multiple proxies to construct a LBM. By sharing overhead of processing packets amongst multiple HIP proxies, a LBM can be more scalable and failure tolerant.

4.1. LBMs adopting Load Balancers

Load balancers have been widely utilized in the design of LBMs. When adopted in a HIP proxy LBM, a load balancer needs to pool all proxy resources and be located in a position where it can intercept DNS lookups or modify DNS lookups if necessary. Also, the load balancer needs to distribute the information it learned from the DNS lookups to the appropriate proxies it manages. Therefore, a load balancer can be regarded as a DNS lookup inspector which distributes overheads

to different DI proxy components according to certain policies. The policies adopted by a load balancer can be various. For instance, a load balancer may require all the packets from a LH be processed by a same HIP proxy while other balancers expect all the packets for a HIP host to be processed by a same HIP proxy.

4.1.1. Load Balancer Supporting DI Proxy Components

In a LBM where a load balancer manages multiple DI-HIT proxy components, the load balancer must be able to intercept, and forward the information about the HIP hosts being queried to the appropriate proxy components. Additionally, the load balancer needs to modify DNS lookup packets and returns HITs in DNS answers to LHs (or resolvers). In order to intercept the packets sent from LHs to HIP hosts, the load balancer may need to advertise a route of HITs.

In a LBM where a load balancer manages multiple DI-NAT proxy components, the load balancer must be able to intercept, and forward the information about the HIP hosts being queried to the appropriate proxy components. Additionally, the load balancer needs to modify DNS answers and returns IP addresses in the address pools of the assigned DI-NAT proxies in DNS answers to LHs (or resolvers). DI-NAT proxies can advertise the routes of the IP addresses in the pools so that the load balancer does not have to intercept the packets between LHs and HIP hosts.

In a LBM where a load balancer manages multiple DI-transparent proxy components, the load balancer must be able to intercept, and forward the information about the HIP hosts being queried to the appropriate proxy components. The load balancer does not modify DNS answers, but it needs to be located in a place(e.g., the egress of the private network) where it is able to intercept the packets sent to HIP hosts and forward them to the assigned proxies.

4.1.2. Load Balancer Supporting N-DI Proxies

When the HIP proxies that a load balancer manages are N-DI proxies, the load balancer must be able to intercept and modify DNS lookups packets. Additionally, the load balancer must be located in a place (e.g., the egress of the private network) where it is able to intercept the packets sent to HIP hosts and forward them to the appropriate proxies. In this solution, the load balancer does not forward the information about the HIP hosts being queried to the appropriate proxies. The N-DI proxies need to consult resolution systems themselves.

4.2. LBMs without Load Balancers

Generally, in a LBM without a load balancer, there are two methods to distribute communication between LHs and HIP hosts among different HIP proxies. The first solution is to divide the LHs in the private network into different groups (e.g., according to their IP addresses), and the LHs in different sections are taken in charge of by different HIP proxies. The second solution is to divide the HIP hosts in the Internet into multiple groups (e.g., according to their HITs or IP addresses), every HIP proxy serves all the LHs in the private network but only take in charge of the packets to and from the HIP hosts in a group. Abstractly, the two solutions are identical. However, the first solution requires a private network to be divided into multiple sub-networks, and each of them is served by a HIP proxy. This may introduce additional modification to the topology of the private network, which is not desired in many cases. Therefore, in the design of existing LBM solutions, the second type of solution can be more preferred. In the remainder of this document, we mainly consider the second one.

4.2.1. Issues Caused by Intercepting DNS Lookups

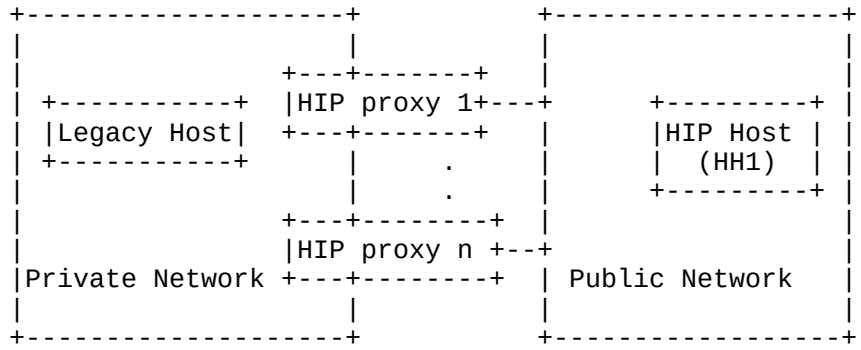


Figure 1: An example of LBM

Figure 1 illustrates a simple LBM. In this mechanism, n proxies are deployed at the border of a private network. If such proxies are DI-HIT proxies, in order to share the overheads in processing data packets, each proxy needs to advertise a route of the HIT section it takes in charge of. In addition, each proxy also needs to advise a route of a section of IP addresses (or a default route for the entire IP address namespace) inside the private network to intercept DNS lookups. A problem occurs when the DNS lookups and the data packets sent by a legacy host are intercepted by different proxies. In such a case, the proxy intercepting a data packet will lack essential knowledge to correctly process it. If the proxies adopted in Figure 1 are DI-transparent proxies, then each proxy only needs to advertise

a route of a section of IP addresses which is adopted to intercept both DNS lookups and data packets. On this occasion, if a HIP host and the DNS server maintaining its RR fall into two different IP sections, the DI-transparent proxy intercepting the lookups for the HIP host will not be the one intercepting subsequent data packets.

A candidate solution to the problem that DI-HIT-proxy-based LBMs and DI-transparent-proxy-based LBMs face is to propagate the mapping information obtained from DNS lookups amongst HIP proxies. Therefore, after intercepting a DNS conversation, a proxy can forward the gained information to the proxy expected to process the subseq uent data packets. Alternatively, a proxy can attempt to collect required information from resolution systems after intercepting a data packet. This approach, however, imposes addition overheads to DI-proxies in communicating with resolution servers.

If the proxies in Figure 1 are DI-NAT proxies, the problem can be eliminated. In a DI-NAT-proxy-based LBM, each DI-NAT proxy needs to advertise two routes, one of the IP addresses in the pool and one of a section of IP addresses for intercepting DNS lookups. After intercepting a DNS lookup, a DI-NAT proxy will return an IP address within the pool in the answer to the requester (a LH or a resolver), which can ensure the subsequent data packets will be transported to the same proxy.

If a DNS resolver supporting DI proxies can forward the mapping information obtained from DNS lookups to appropriate HIP proxies, the issue can be easily addressed. In this case, the DNS resolver actually acts as a load balancer.

4.2.2. Issues with LBMs in Capturing and Processing Replies from HIP hosts

Theoretically, when representing a LH to communicate with a HIP host in the public network, a HIP proxy can use either an IP address it possesses or the IP address of the LH as the source address of the packets forwarded to the HIP host. However, in practice, the later option may cause an asymmetric traffic issue in the load balancing scenarios where multiple HIP proxies provide services for a same group of LHs. Assume there are two HIP proxies located at the border of a private network. If the proxies adopt the later solution, they need to advertise the routes of the LHs in the public network respectively. As a result, it is difficult to guarantee the packets transported between a legacy host and a HIP host are stuck to a same HIP proxy, and thus after a proxy intercepts a packet it may lack the proper HIP association to process it.

A possible solution to address this problem is to share HIP state

information (e.g., HIP associations, sequence number of IPsec packets) amongst the related HIP proxies in a real-time fashion. However, during communication, some context information such as the sequence numbers of IPsec packets can change very fast. It is infeasible to synchronize the IPsec message counters for every transmitted or received IPsec packet, since such operations will occupy large amounts of bandwidth and seriously affect the performances of HIP proxies. [Nir 2009] indicates that this issue can be partially mitigated by synchronizing IPsec message counters only at regular intervals, for instance, every 10,000 packets.

An issue similar with the one mentioned above is discussed in [TSC05], and an extended HIP base exchange is proposed. But the proposed solution only tries to help HIP-aware middle boxes obtain the SPIs used in a HIP base exchange and cannot be directly used to address the issue mentioned above.

When adopting the preceding option, proxies need to advertise the routes to their addresses in the public network respectively, and so the packets transported between a LH and a HIP host are intercepted by the same proxy. The issue discussed above can thus be addressed. In the following discussions, without mentioning otherwise we assume that a HIP proxy uses one of its IP addresses as the source IP address of a packet which it sends to a HIP host.

5. Issues with LBMs which also Support HIP Hosts to Initiate Communication

Apart from the basic functions (i.e., supporting LHs to communicate with HIP hosts), in many typical scenarios, HIP proxies may also need to facilitate the communication initiated by HIP hosts. In this section, we attempt to analyze the issues that a HIP proxy has to face in the conditions where HIP hosts proactively initiate communication with legacy hosts.

In order to support the communication initiated by HIP hosts, the HIP proxies of a private network should have the knowledge essential to represent the LHs to perform HIP BEXs. Such knowledge consists of the IP addresses of the legacy hosts, their pre-assigned HITs, the corresponding HI key pairs, and any other necessary information. In addition, such information of the LHs should be advertised in resolution systems (e.g., DNS and DHT) as HIP hosts. Otherwise, a HIP host has to obtain the HIT of the LH in the opportunistic model which, however, should only be adopted in secure environments.

5.1. DNS Resource Records for ML Hosts

In practice, the AAAA RR of a LH can consist of either the IP address of the LH or the address of its HIP proxy. In the preceding approach, the routing infrastructure will try to forward the packets for the LH to the host directly. Therefore, in this case, HIP proxies must be located on the path of such packets to intercept them. In the later approach, the packets for a legacy host are firstly forwarded to the associated HIP proxy. Compared with the preceeding approach, the later case enable a proxy then to be deployed more flexibly and to be more efficient in private networks where legacy hosts and HIP hosts are deployed in an intermixed way, since the HIP proxy will not have to intercept the packets transported between HIP hosts. However, the later approach may cause problems when processing packets sent by legacy hosts in the public network. Normally, a HIP proxy needs to serve a number of LHs. When using the later approach, the packets destined to these LHs will have a same destination address (i.e., the IP address of the proxy). Therefore, when receiving a packet from a legacy host located in the public network, the proxy may find it difficult to identify the LH which the packet should be forwarded to.

A simple approach which combines the advantages of the above two solutions but avoids their disadvantages is to extend the RDATA field in HIP RRs [RFC5205] with a new proxy field, which contains the IP address of a HIP proxy. In the extended HIP RR of a LH, the proxy field consists of the IP address of its HIP proxy, while the proxy field in the RR of an ordinary HIP host is left empty. Therefore, a HIP host intending to communicate with the LH can obtain the IP address of the proxy from DNS servers and set it as the destination address of the packets. The packets are then routed to the proxy. When a non-HIP host intends to communicate with the legacy host, it can obtain the IP address of the legacy host from the AAAA RR as usual and set it as the destination address of the packets; the packets are then transported to legacy host directly.

It is also possible to use the RVS field in a HIP RR to transport the information of a HIP proxy. However, in certain scenarios, a special proxy field can bring additional benefit in security. For instance, it is normally assumed that the BEX protocol is able to establish a security channel for the hosts on the both sides of communication to securely exchange messages. However, this presumption may be no longer valid in the presence of HIP proxies, as the messages between legacy hosts and proxies can be transported in plain text. With the Proxy field, it is easy to distinguish the legacy hosts made up by HIP proxies from the ordinary HIP hosts. Therefore, a HIP host can assess the risks of exchanging sensitive information with its communicating peers in a more precise way.

5.2. An Asymmetric Path Issue

In a load balancing scenario where multiple HIP proxies are deployed at the border of a private network, the packets transported between a legacy host and a HIP host may be routed via different HIP proxies. Therefore, when a packet is intercepted by a HIP proxy, the proxy may find that it lacks essential knowledge to appropriately process the packet. Hence, an asymmetric path issue occurs.

In order to explain the asymmetric path issue in more detail, let us revisit the LBM illustrated in Figure 1. In addition, assume that the HIP proxies are DI-HIT proxies and their IP addresses are maintained in the DNS RRs of the LHs. When a HIP host (e.g., HH1) looks up a legacy host at a DNS server, the DNS server returns the IP addresses of all the HIP proxies in an answer (see Figure 2). Upon receiving the answer, HH1 needs to select an IP address and sends an I1 packet to the associated HIP proxy. Assume the HIP proxy 1 is selected. Then after a base exchange, HIP proxy1 and HH1 establish a HIP association respectively. Upon receiving the first data packet from HH1, the HIP proxy uses the HIP association to de-capsulate the packet and forwards it to the legacy host. In the forwarded packets, the HIT of HH1 is adopted as the source IP address, and thus the HIT of HHI is adopted as the destination address in the reply packets sent by the legacy host. Assume that the HIT of HH1 is within the section managed by HIP proxy n. According the routes advertised by the proxy n, the packet is forwarded to the HIP proxy n which, however, does not have the corresponding HIP association to deal with the packet. Similarly with DI-HIT proxies, DI-transparent proxies and N-DI proxies also suffer from the asymmetric path issue in the load balancing scenarios, since they cannot guarantee the data packets which are transported between a legacy host and a HIP host stick to a single HIP proxy too.

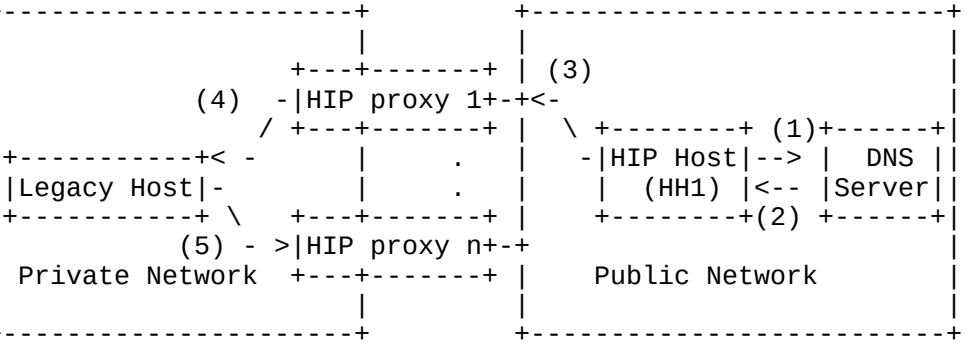


Figure 2. An example of the asymmetric path issue

As we mentioned in section 3.3.1, the approach of synchronizing HIP associations and IPsec associations amongst HIP proxies can be used

to address this issue. However, this issue will introduce additional communication overhead on HIP proxies. Here, we discuss several other alternative solutions.

The simplest solution is to allow a HIP proxy to discard the I1 packets it receives if they are not original from HIP hosts which the proxy takes in charge of. In addition, the proxy can inform the senders of the incidents using ICMP packets. Therefore, after waiting for a certain period or upon receiving a ICMP packet, a HIP host will try to select another HIP proxy from the list in the DNS answer and send an I1 packet it. In the worst case, this process needs to be recursive until all the HIP proxies in the list have been contacted. Because a HIP host may have to send the multiple I1 packets in order to communicate with a LH, this solution may yield a long delay. Note that in some DNS based load balancing approaches, a DNS server only returns one HIP proxy in an answer. On such occasions, HIP hosts have to communicate with DNS servers repeatedly, and the negative influence caused by the communication delay can be even exacerbated.

A solution which is able to avoid the delay issue is to endow DNS servers with the capability of returning the IP address of an appropriate HIP proxy in an answer according to the HIT (if the proxy is a DI-HIT proxy or a N-DI proxy) or the IP address (if the proxy is a DI-transparent proxy) of a requester. That is, the HIP proxy described in a DNS answer should take in charge of the namespace section which the requester belongs to. In order to achieve this, DNS servers need to 1) maintain the information about the sections of the namespaces that HIP proxies take in charge of, 2) locate the appropriate HIP proxy according to the HIT or the IP address of a HIP requester. These requirements result in modifications to current DNS servers in the implementation of the DNS server applications and the conversation protocols between requesters and DNS servers. For instance, a HIP host may need to transport its HIT in DNS requests in order to help DNS servers locate an appropriate HIP proxy. An negative impact of this solution is to introduce additional complexity and overhead to DNS servers.

Another solution is to extend RVS servers as load balancers. After receiving an I1 packet from a HIP host, the load balancer then select an proper HIP proxy and forward the packet to it. Using this solution, a DNS server only needs to reply a record on receiving a query from a HIP host, which reduce the traffic transported between DNS servers and HIP hosts.

The asymmetric path issue can be eliminated when DI-NAT proxies are adopted. A DI-NAT proxy located at the border of a private network maintains a pool of IP addresses which are routable in the private

network. After receiving a packet from a HIP host, the DI-NAT proxy processes the packet and forwards it to the destination legacy host. In addition, an IP address selected from the pool is adopted as the source address of the packet. Therefore, when the legacy host sends responding packets to the HIP host, the packets will be transported to the same HIP proxy. The asymmetric path issue is thus eliminated.

6. Issues with LBMs in Supporting Dynamic Load Balance and Redundancy

In practice, there are requirements for LBMs to support dynamic load balance and redundancy. That is, when a proxy in a LBM is not able to work properly or the overheads imposed on it surpass a threshold, the proxy can delegate all of (or a part of) its job to other proxies. A proxy provide backup sevice is called a backup proxy, and the proxy served by a backup proxy is called a primary proxy. Note that two proxies can be backup proxies for each other on different jobs. In this section, we analyze the performance of different types of HIP proxies in supporting dynamic load balance and redundancy.

If there is a load balancer intercepting and distributing traffic among different proxies, the balancer can flexibly forward traffic to other proxies when a proxy cannot work properly. However, if there is no load balancer deployed, in order to provide backup services, a backup proxy has to advertise the same routes as those advertised by the primary proxy in both the private and the public networks. To avoid affecting the normal operations of the primary proxy, the routes advertised by the backup proxy have a much higher cost than that of the routes advertised by the primary proxy. When the abnormal conditions mentioned above occurs, the primary proxy can withdraw the routes it previously advertised so that the packets supposed to be processed by the primary proxy will be forwarded to the backup proxy. We refer to the routes advertised by a proxy for backup purposes as the backup routes of the proxy. In contrast, we refer to the routes advertised by a proxy to achieve its primary job as the primary routes of the proxy. In practice, the proxies in a LBM can provide backup services for one another. Therefore, a proxy in such a LBM may needs to advertise both primary and backup routes.

The synchronization of state information between primary and backup proxies is also very important. Without proper HIP associations, a backup proxy cannot correctly take place of the primary proxy to process the packets. The state synchronization problem has been discussed above. If there is no state synchronization, a backup proxy may select to send signaling packets to HIP hosts to initiate new HIP BEXs.

In the remainder of this section, we discuss the operations of

different types of HIP proxies in achieving dynamic load balance and redundancy without the assistance of load balancer.

6.1. Application of DI-HIT proxies in supporting dynamic load balance and redundancy

As mentioned in section 3.1, a DI-HIT proxy needs to at least advertise two primary routes in the private network, a route of a section of HITs for intercepting data packets, and a route of a section of IP addresses for intercepting DNS lookups. When the proxy cannot work properly, it can withdraw both routes to enable a backup proxy to take over its job.

In some cases, a DI-HIT proxy may only want to delegate a part of its job to others so as to reduce the overloads it undertakes. To achieve this objective, the proxy can divide its routes into multipe more detailed routes. When the overload on the proxy is high, it can only withdraw a subset of the routes. For instance, a DI-HIT proxy can selectively only delegate a part of the responsibility in processing DNS lookups to a backup proxy by withdrawing one of its lookup intercepting routes.

6.2. Application of DI-NAT proxies in supporting dynamic load balance and redundancy

A DI-NAT proxy needs to at least advertise two primary routes in the private network, a route for its IP address pool, used to intercept data packets, and a route for an IP address section used to intercept DNS lookups. When the proxy cannot work properly, it can withdraw both routes to enable a backup proxy to take over its job. In this case, the delegated backup proxy needs to maintain an IP address pool identical to the one maintained by the primary proxy. Moreover, apart from synchronizing HIP associations, the synchronization of mappings from IP addresses to HITs is also required. Otherwise, the backup proxy cannot translate the received packet correctly.

If a DI-NAT proxy only intends to maintain existing communication between LHs and HIP hosts while not facilitating any more, it can withdraw the lookup intercepting route. As mentioned previously, DI-NAT proxies have the capability to stick the DNS lookups and the subsequent data packets to the same proxy. Therefore, the backup proxy can intercept DNS lookups as well as process the subsequent communication.

6.3. Application of DI-transparent proxies in supporting dynamic load balance and redundancy

Unlike DI-HIT and DI-NAT proxies, the routes advertised by a DI-

transparent proxy are used for intercepting both DNS lookups and data packets. Therefore, before a DI-transparent proxy withdraws a route, it needs to synchronize the states of the on-going communication affected by the routing adjustment to its backup proxies.

7. Conclusions

This document mainly analyzes and compares the performance of different kinds of HIP proxies in LBMs. Amongst the HIP proxies discussed in the document, DI-NAT proxies show its advantages in multiple scenarios. In addition, we argue that the state synchronization among HIP proxies is very important to achieve load balancing and redundancy. There is a topic which is important but not covered in this document is the compatibility among different HIP proxies. The different types of HIP proxies are designed based on different presumptions. The presumptions of different type of HIP proxies maybe conflict with each other. Then how to make a trade-off and enable different types of proxies work cooperatively is an important issue that the designers of HIP extensible solutions have to consider.

8. IANA Considerations

This document makes no request of IANA.

9. Security Considerations

One design objective of HIP is to provide peer-to-peer security between communicating hosts. However, when a HIP host communicates with a LH under the assistance of a HIP proxy, the security of the communication between the HIP proxy and the LH may not be protected. If the HIP proxy is transparent to the HIP host, the host will believe that it is communicating with a ordinary HIP host and will not realize that the peer-to-peer security between it and the LH is not guaranteed. This may cause potential security risks, especially when the HIP proxy is located in the public network. Therefore, some solutions should be provided for a HIP hosts to detect whether they are actually communicating with HIP proxies.

When sharing HIP state information amongst HIP proxies, the integrity and confidentiality of the state information should be protected. The discussion about the similar issues can be found in [Nir 2009] and [Narayanan 07].

If a HIP proxy is deployed at the border of a private network or

within the boundary of a private nework, the security issues with the communcation between the proxy and LHs are not serious. However, if a proxy is deployed in the public network, both the communication between LHs and the proxy and the communication between the proxy and DNS servers should be secured.

10. Acknowledgements

Thanks Thomas.R.Henderson for his kindly prove-reading and precious comments.

11. References

11.1. Normative References

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

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.

[RFC5205] Nikander, P. and J. Laganier, "Host Identity Protocol (HIP) Domain Name System (DNS) Extensions", RFC 5205, April 2008.

[RFC5338] Henderson, T., Nikander, P., and M. Komu, "Using the Host Identity Protocol with Legacy Applications", RFC 5338, September 2008.

11.2. Informative References

[Narayanan 07] Narayanan, V., "IPsec Gateway Failover and Redundancy - Problem Statement and Goals", 2007.

[Nir 2009] Nir, Y., "IPsec High Availability Problem Statement", 2009.

[PAT07] Salmela, P., Wall, J., and P. Jokela, "Addressing Method and Method and Apparatus for Establishing Host Identity Protocol (Hip) Connections Between Legacy and Hip Nodes, US. 20070274312", 2007.

[SAL05] Salmela, P., "Host Identity Protocol proxy in a 3G

system", 2005.

[TSC05] Tschofenig, H., Gurtov, A., Ylitalo, J., Nagarajan, A., and M. Shanmugam, "Traversing Middleboxes with the Host Identity Protocol", 2005.

Authors' Addresses

Dacheng Zhang
Huawei Technologies Co.,Ltd
HuaWei Building, No.3 Xinxi Rd., Shang-Di Information Industry Base, Hai-Dian District
Beijing, 100085
P. R. China

Phone:
Fax:
Email: zhangdacheng@huawei.com
URI:

Xiaohu Xu
Huawei Technologies Co.,Ltd
HuaWei Building, No.3 Xinxi Rd., Shang-Di Information Industry Base, Hai-Dian District
Beijing, 100085
P. R. China

Phone:
Fax:
Email: xuxh@huawei.com
URI:

Jiankang Yao
CNNIC
4, South 4th Street, Zhongguancun
Beijing, 100190
P.R. China

Phone:
Fax:
Email: shenshuo@cnnic.cn
URI:

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 7, 2011

D. Zhang
Huawei Technologies Co.,Ltd
D. Kuptsov
HIIT
S. Shen
CNNIC
March 6, 2011

Host Identifier Revocation in HIP
draft-irtf-hiprg-revocation-02

Abstract

This document mainly analyzes the key revocation issue with host identities (HIs) in the Host Identity Protocol (HIP). Generally, key revocation is an important functionality of key management systems; it is concerned with the issues of removing antique cryptographic keys from operational usages when they are not secure or not secure enough any more. This functionality is particularly important for the security systems expected to execute for long periods. This document also attempts to investigate several key issues that a designer of HI revocation mechanisms need to carefully consider.

Requirements Language

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 [RFC2119].

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 7, 2011.

Copyright Notice

Table of Contents

1. Introduction

In a HIP architecture [RFC5201], a HIP host needs to generate a public key pair before it communicates with other HIP hosts. The public key is used as its HI while the private key is kept securely by the host. When two HIP hosts attempt to initiate a conversation (e.g., a TCP session), they can take advantage of their HI key pairs to perform mutual authentication and generate keying materials for securing subsequent data and signaling packets. Therefore, the security of HIP architectures largely relies on the security of those HI key pairs. If the HI key pair of a HIP host is revealed, an attacker can easily impersonate the victim to carry out malicious attacks without being detected.

It has been widely recognized that a cryptographic key (which can be either a symmetric key or a public key) should have a reasonable valid period [Recommendations]. After having been employed for a certain period, a cryptographic key will be in more dangers of compromise. As time elapses, an attacker can collect more materials (e.g., encrypted data, signatures and associated plain texts, etc.) and obtain more time to compromise the key. In addition, unexpected key disclosure is a common practical issue, which may be caused by, e.g., improper key management policies or hardware stealing. Consequently, in the design of a security system which is expected to execute for a long period, the issues with revoking the cryptographic keys which do not have enough security strengths must be considered.

In current HIP architectures, the key revocation issues with transient (session) keys have been well discussed. HIP allows two communicating hosts to update their transient keys securely at run time. However, the key revocation issues with permanent keys (i.e., HIs) have not been well explored yet. No facility is provided for HI revocation either.

2. Terminology

BEX: Base Exchange

HIP: Host Identity Protocol

PKI: Public Key Infrastructure

3. Key Management

Key management aims at guaranteeing the security of cryptographic keys during the period of their application and includes all of the

provisions made in a security system design which are related to generation, validation, exchange, storage, safeguard, application, and replacement of cryptographic keys. Appropriate key management is critical to security mechanisms providing confidentiality, entity authentication, data origin authentication, data integrity, and digital signatures. Specifically, a full-fledged key management system should be able to support [Menezes et al. 1996]:

1. Initialization of system users within a domain;

2. Generation, distribution, and installation of keying material;

3. Controlling the use of keying material;

4. Update, revocation, and destruction of keying material; and

5. Storage, backup/recovery, and archival of keying material.

4. Key Revocation

Key revocation is an essential functionality of a security system. By refreshing antique cryptographic keys, a security system can reduce the dangers of being compromised. Key revocation is also an important step when a security system attempts to confine and recover from the damages caused by attacks. The criteria measuring a key revocation mechanism should include security, efficiency, latency, overheads in terms of communication, etc.

4.1. Classification of permanent Key Revocation Mechanisms

Cryptographic keys adopted in a security system can be classified into permanent keys and transient keys according to their life periods. As indicated by the name, permanent keys are maintained by holders for relatively long periods which can be various from months to years. Because frequent usages of permanent keys can damage their security strength and reduce their valid periods, in many security mechanisms, permanent keys are employed to generate and distribute transient keys which are only valid in relatively short periods (e.g., within a single TCP session). Key revocation issues with transient keys have been taken account of in most authentication mechanisms (e.g., Kerberos, IPSec, SSL, etc.). For instance, in Kerberos, a user can use her password to obtain a session key from a KDC; the session key then can be further used to securely discard and update old sub-session keys. The revocation of transient keys is also considered in the design of HIP. A basic handshaking protocol (i.e., the HIP Base Exchange) has been specified. Using it, two communicating HIP hosts can employ the authenticated Diffie-Hellman

algorithm to securely distribute keying materials which will be used to generate new cryptographic keys in the following communication. After a handshake, the hosts are able to refresh their transient keys and the corresponding HIP associations, using Update packets.

The revocation issues with permanent keys are also taken into account in lots of key management mechanisms (e.g., PGP, PKI, Peer-to-Peer Key Management for Mobile Ad Hoc Networks [Merwe et al. 2007]). Particularly, in PKI, key revocation issues are addressed in certificate revocation mechanisms.

4.2. Classification of permanent Key Revocation Mechanisms

This draft focuses on the issues with permanent key revocation in HIP. In the remainder of this draft, key revocation indicates permanent key revocation, without mentioning otherwise.

Mechanisms for key revocation can be classified in various ways, according to:

- Whether additional operations are needed. If a key revocation mechanism does not need any additional operation in the revoking process of a cryptographic key, it is called an implicit key revocation mechanism. The basic idea of an implicit HI revocation mechanism is to associate a key with a valid period and use cryptographic methods to prove the binding between the key and its valid period. Therefore, after the pre-defined period expires, the key is obsolete automatically. For instance, in PKI, a Certificate Authority (CA) can issue a certificate for a user in order to assert the association between the user and its public key. The certificate is associated with a life period. When the period expires, the user's public key is revoked automatically. If a key revocation mechanism needs to carry out additional operations (e.g., notifications) to revoke a cryptographic key, it is called an explicit key revocation mechanism. In different explicit key revocation mechanisms, such operations can be performed either by a dedicated server or by the owner of the key. Compared with implicit key revocation mechanisms, an explicit key revocation mechanism has the capability to revoke a cryptographic key before its life period expires. For instance, in X.509 [RFC2459] based systems, an issuer can generate a list of certificates, which were revoked for some reasons before their expiring dates, for users to consult.

- Whether a secure third party is needed. In some revocation mechanisms, the status information of a cryptographic key is provided by a secure third party. A proof of validity is performed during each request from users, and the secure third

party provides up-to-date information. Online Certificate Status Protocol (OCSP) for X.509 certificate is such a mechanism. An OSPF client generates an OCSP request that primarily contains the information of one or more queried certificates and send it to a trusted OCSP server. After receiving the OCSP request, the server creates an OCSP response containing the updated status information of the queried certificates. In some other revocation mechanism, validity information is distributed to the requester by a non-secured server. For example, in PGP, a principal can use its revoked key to sign a key revocation certificate and uploaded it to a key repository server. The server is regarded as "non-secured" only because the server only provides a repository service and does not make any assertion. Certificates themselves are individually secured by the signatures thereon, and need not be transferred over secured channels. In fact, authorization policies to a repository server in the form of write and delete protection is mandatory so as to enable maintenance and update without denial of service.

- The list is adopted. According to the information provided, key revocation mechanisms can be classified into black list mechanisms and white list mechanisms. A black list mechanism can provide the information of the keys which are not valid anymore. The Certificate Revocation List (CRL) is an example of this kind of mechanism. In a CRL, revoked certificates are listed in a signed list, so that users can query the information about the revoked keys whenever it is convenient. White list mechanisms, instead, only provide information of valid keys. For example, SSH specify a kind of resource record (RR) called SSHFP [RFC4255]. A SSHFP RR contains the information of the fingerprint of a valid cryptographic key. If a key needs to be revoked, the associated SSHFP RR is removed. If a user cannot find the associated SSHFP RR from DNS, she will believe that the key inquired about is no longer valid.

- The way of distributing revocation information. In a key revocation mechanism applying the push model, when a key is revoked, a server proactively contacts the related users to inform the case. In contrast, in a key revocation mechanism applying the pull model, a client needs to query a server for particular revocation information. OCSP, CRL, and the key revocation mechanisms adopted in PGP and SSH all belong to this category.

There are few discussions about the HI revocation issues with HIP. In the current HIP architecture, hosts are allowed to update their identifiers arbitrarily without notifying others. The lack of HI revocation mechanism can be taken advantage of by attackers to, for instance, escape tracking, bypass ACLs (Access Control Lists),

impersonate others using the compromised HIs, etc. In remainder of this document, candidate approaches and related issues are discussed.

5. Implicit HI Revocation in HIP

Implicit key revocation is the simplest key revocation approach. By associating an HI with a life period, the holder of the HI needs to update the HI periodically so as to reduce the risk that the HI is compromised. In addition, life periods of HIs can help users to verify how long an HI has been used and how long the HI will still be valid. This enables host managers to define more specific security policies.

Note that the HI and the HIT of a host are cryptographically associated. A revocation of an HI will cause the revocation of the corresponding HIT, and vice versa. The life periods of an HI and its HIT are identical; the revocation of a HI implies the revocation of the associated HIT, and vice versa.

The life period of an HI can be specified either by the holder of the HI or by a trusted authority. During HIP BEXs, such life period information can be encapsulated in parameters and transported within HIP packets. If the life period of the HI is specified by its holder, the holder needs to use the associated private key to sign the parameter. If the life period of the HI is specified by a trusted authority, the authority needs to use its private key to sign a life period certificate for the HI. The certificate can be encapsulated within a CERT parameter and transported in HIP packets.

Figure 1 illustrates an extended HOST_ID parameter which is able to transport an HI and the associated life period. This parameter can be adopted in the cases where the life period of the HI is specified by its holder. Similar to the live periods of X.509 certificates, the life period of an HI is specified by a Not Before Time and a Not After Time. In this parameter, the NB Length and NA Length fields indicate the lengths of Not Before Time and Not After Time fields respectively. The Not-Before-Time and the Not-After-Time can be in a format of either UTCTime or GeneralizedTime defined in [RFC2459].

During a HIP base exchange, the parameter containing Initiator’ s HI and the associated life period information is transported in the I2 packet, while the parameter containing Responser’ s HI and the associated life period information is transported in the R1 packet.

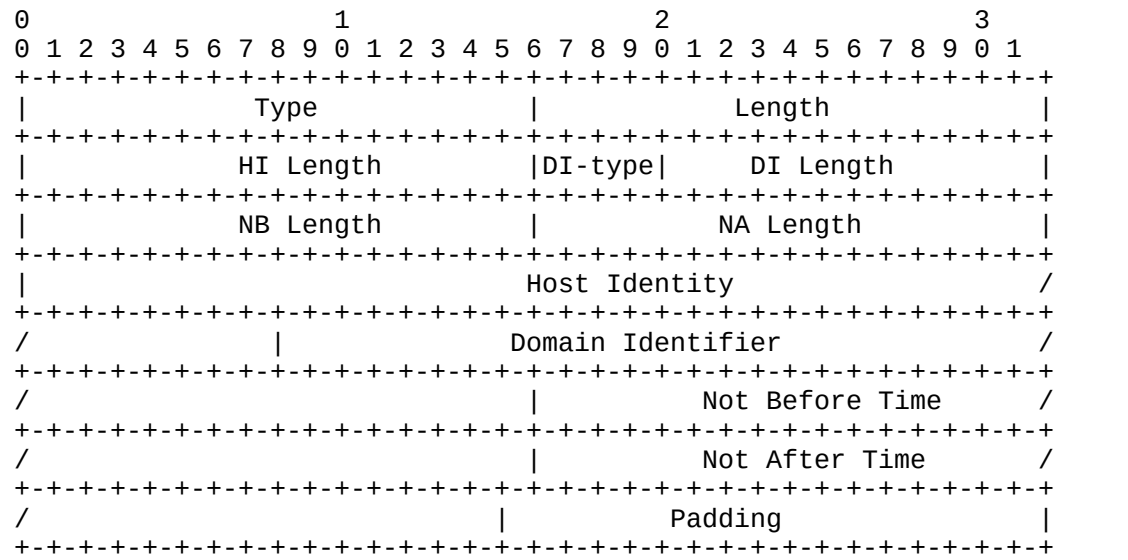


Figure 1. An extension of HOST_ID parameter

The approach to enabling a holder to specify the life period of its HI does not rely on any dedicated trusted authority and introduces little performance penalty in verifying the life period. However, a concern about this approach is how to ensure that HIP hosts will appropriately define and manage the life periods of their HIs. In practice, the revocation and refreshment of an HI can be quite complex. Apart from updating the key material locally, additional operations also need to be performed (e.g., updating the associated HIP resource record in DNS, proactively informing the partners which may be affected by the revocation, etc.). Therefore, a lazy manager of a HIP host may attempt to avoid refreshing the HI and HIT of her host. If the manager assigns an extremely long life period for its HI, other HIP hosts can easily detect the problem and refuse to communicate with the host. However, if the manager selects to assign a new life period with a reasonable length for her HI piror to the expiration of the old life period, the renewal of the life period can be difficult to be detected in current HIP architectures. In practice a HIP host normally does not maintain the HIs and other related information of its communicating partners for a long period, in order to reduce memory consumption and foil deny-of-service attacks. Moreover, because HITs are treated by applications as ordinary IP addresses which have no expiration date, in referral scenarios the receiver of a HIT may not be able to obtain the knowledge of the life period of a HIT from the referrer. In the current HIP resolution solutions (e.g., HIP RR), there is no concern about the life periods of HIs. On such occasions, a host can only obtain the life period information from its communicating partner

(i.e., the holder of the HI). Therefore, in current HIP architectures, the approach which allows a holder to specify the life period of its HI can only be feasible in the environments where there has already been a certain level of trust between two HIP hosts beforehand, that is, a HIP host can believe its communicating partner has specified an appropriate life period for its HI and will only attempt to use it within the valid period.

The issues mentioned above can be largely addressed by assigning a trusted authority to manage the life periods of HIs. However, a dedicated trusted third party may introduce complexity into the current HIP architecture, impose additional communications (e.g., registration process, generation of certificate chain, etc.), and cause issues in terms of scalability and trust. The details of the issues imposed by such dedicated authorities are discussed in section 6.

In the remainder of this sub-section, we introduce two complementary approaches to mitigate the issues of arbitrarily modifying HI life periods while imposing little performance penalty to HIP hosts. The first approach is to extend resolution systems (e.g., DNS servers) to provide trustable life-period information of HIs. In this approach, the life-period information can be encapsulated in the same packet with other mapping information and sent back to users so as to eliminate additional communication between users and resolutions systems.

In order to achieve this, space for the life period information needs to be allocated in the resource records sent back to users. In Figure 2, an extension of the HIP RR with life period information is illustrated. Same as the extended HOST_ID parameter in Figure 1, the NB Length and NA Length fields indicate the lengths of Not Before Time and Not After Time fields respectively. The Not-Before-Time and the Not-After-Time can be in a format of either UTCTime or GeneralizedTime defined in [RFC2459].

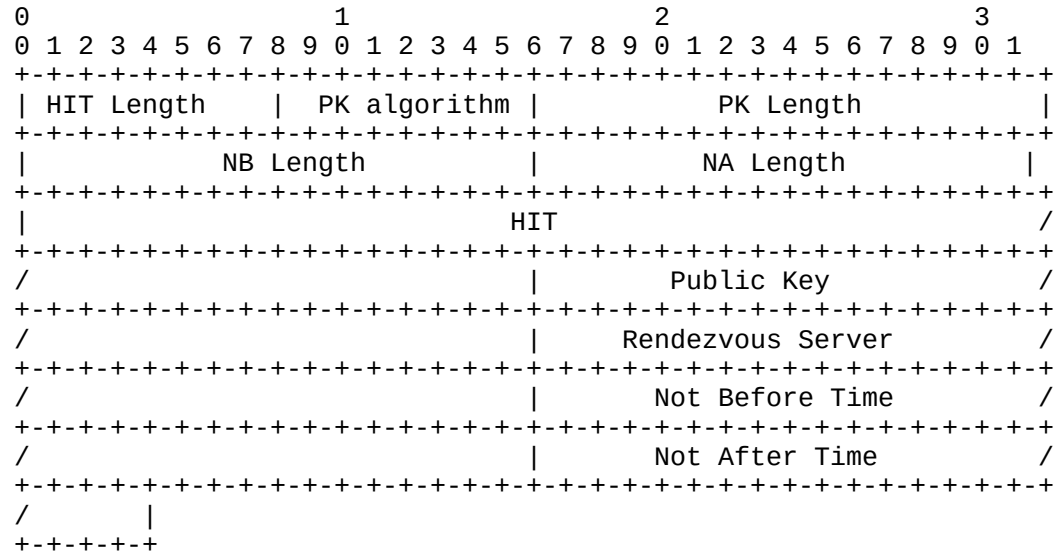


Figure 2. An Extension of HIP RR

The basic functionality of a resolution server is to provide mapping information for users. In practice, it is normally the responsibility of authorized users to maintain and update the contents of RRs while resolution severs can verify the contents of RRs against certain security policies. Therefore, in this approach, information of the life period of an HI, just like the other information in the RR, can be provided by an authorized user at the registration time. After the registration, the life period information is only allowed to be updated by the ones who have higher privileges (e.g., server managers).

Let us use DNS servers as an example. After a user uploads the information of a HIP host in an authoritative DNS server, the user is not allowed to modify the Not Before Time and Not After Time fields of the HI any more. Moreover, after the life period of the HI has expired, the associated RRs needs to be removed.

Until now, the ID to Locator mapping solution in HIP has not been standardized yet. We argue that it is desired to integrate the implicit key revocation functionality into such systems.

The second approach is to introduce the life period of a HI into the generating process of the associated HIT. For instance, the life period of an HI can be used as a part of the input for generating the associated HIT. Therefore it is computationally difficult even for the holder of the HI to modify the life period without modifying the HIT. Therefore, after a host advertises its contacting information

in resolution servers, any attempts to modify the life period of the HI can be easily detected. For instance, in the case that a host obtains a HIT from its referrer, it needs to first obtain the knowledge to access the host holding the HIT from resolution servers. Then it can get the associated HI and the life period from the HIT holder, and re-calculate the HIT to verify whether the life period of the HIT is valid. This approach needs little modification on the resolution servers and can be applied independently. A disadvantage of this approach is its inflexibility in the cases where the life periods of HIs need to be extended.

6. Explicit HI Revocation in HIP

As mentioned previously, in many typical scenarios a cryptographic key should not be used any more even when it is still in its valid life period. For instance, when a key is detected to be compromised, it must to be revoked immediately even if it has not reached its expiration date. In such a case, explicit key revocation is needed.

When an HI needs to be removed from operational use prior to its originally scheduled expiry, the revocation of the HI needs to be informed to all the hosts which might be affected. If there is no dedicated third party to rely on, the holder of the HI needs to deliver the revocation certificate signed by the associated private key to all the affected partners. The poor scalability of this type of solution is always a subject of debates. First, this solution requires the holder an HI to maintain a long list of information about the partners, that may be affected by the revocation; this job can be onerous and error prone. In addition, because HIP does not support multicast, the holder has to generate a notification packet for each of its partners, and send them out during the revocation. When the number of related partners increases, the holder may have to spend a large amount of bandwidth, memory and computing resources in generating and delivering the notification packets. In order to improve the performance of this solution, the holder can send the certificate to a limited set of partners. These partners then relay the certificate to others. However, this solution may introduce additional latency and make the delivery of the certificate un-reliable. Besides the above issues, this solution requires all the involved partners to be online during an HI revocation process, which can be hardly fulfilled on many occasions. Basically, this solution is only suitable in the circumstances where the number of involved hosts is relatively small and stable.

The experiences in PKI demonstrate that pull models can be more scalable in dealing with a large amount of users, and as a result, most of the certification revocation mechanisms (e.g., Certification

Revocation Lists (CRLs), delta CRLs [RFC2459], and the On-Line Certificate Status Protocol (OCSP)) proposed in PKI are based on pull models. In these mechanisms, the revocation information is maintained in a third party for users to query whenever it is convenient.

PKI has provided a set of certificate management mechanisms. On many occasions, it is feasible for HIP to take advantage of PKI style solutions to address the issues with HI management.

However, it should be realized that PKI oriented solutions are not silver bullets and cannot be utilized to address all the issues that HIP has to encounter. After HIP has been globally deployed, it is expected that there will be billions of HIP users which may belong to different organizations and attach to the Internet through different ISPs. Due to the poor scalability of PKI and lack of trust, it is extremely difficult (if possible) to put such a big amount of geographically distributed users under the control of a unique PKI security domain. Therefore, it is reasonable to assume that there will be many different security domains all over the world. When two HIP hosts belong to two different security domains, it may be difficult for a host to verify the assertion made by the security server in the domain of the other one. Although there have been solutions of generating trust relationship across various security domains, all of them impose additional overheads with respect to the construction and verification of credential chain and communication with remote security servers, which negatively influences the performance of HIP. Therefore, the HIP community argues that two HIP-aware hosts should be able to communicate without any additional security facilities. Actually, the only third party server introduced in the base-line HIP architecture is the Rendezvous Server (RVS)[RFC5204]. A RVS only relays messages for the hosts which attempts to communicate with mobile hosts and provides little security functionality. The HIP hosts intending to communicate with each other still need to use the HIP Base Exchange protocol to carry out authentication and exchange keying material for future communications. However, RVSes can be extended to support HI revocation if necessary. When a mobile host changes its HI, it can inform its RVS. Therefore, when the RVS find that a host attempts to access the mobile host with the old HI, the RVS can send the mapping information of the antique HI and the new HI to the host. The RVS needs to use its private key to sign the mapping information in order to ensure the information will not be tampered with. Upon receiving the mapping information, the remote host can use the new HI in the subsequent communications. Additionally, since it is suggested in [RFC5204] that a user get the information of RVSes from DNS, the security of the communication between the remote host and DNS servers needs to be protected. Otherwise, an attacker can easily convince a

witness that she is a legal RVS by forwarding a bogus DNS RR consisting of its information to the witness. DNSSEC can be applied to address this issue.

Also, resolution servers can be potentially adopted to construct a global explicit HI revocation mechanism applying a pull model. For instance, when a host intends to revoke its HI, it can send a revocation certificate signed by its private key to an authoritative DNS server. After receiving the certificate, the correspondent RR will be removed, and thus users will not obtain the information about the revoked HI any more. Therefore, DNS servers can perform as a white list HI revocation mechanism, similar to what is specified in SSH. To avoid the long delay in the spread of revocation information caused by caching RRs on DNS resolvers, the TTL (Time To Life) of RRs can be set to zero. In order to secure the revocation information, DNSSEC should be adopted.

7. Related Discussions

7.1. Influence of HI revocation on Already Generated HIP Associations

In this sub-section, we investigate the possibility of using already generated HIP associations to transport the update information after the correspondent HI key pair is no longer valid.

In a BEX, HI key pairs of the both communicating partners are used to carry out mutual authentication while the key materials for securing subsequent communication are generated by the Diffie-Hellman algorithm. Therefore, if an HI key pair is secure at the time when a HIP association is generated, the later revocation of the HI key pair will not affect the security of the keying materials. Assume there is an attacker which has compromised the HI key pair. It is still computationally difficult for the attacker to decrypt the packets transported between the communicating partners. Because the Update packets are under the protection of HMAC, the attacker cannot forge them to interfere with the communications. Note that the attacker can try to forge Notify packets. However, according to [RFC 5201] Notify packets are only informative, which will not affect the state of the communicating partners. Therefore, if no explicit key revocation occurs, the expiry of an HI will not affect the security of HIP associations generated using the HI when it is still valid. They still can be used until they reach their expiring time. However, if an HI is found to be compromised, the security of the keying materials of the already generated HIP associations cannot be guaranteed. In practice, the compromise of a cryptographic key can be perceived only after the attacks employing the key are detected. It is difficult for one to identify the exact time from which the key

is no longer secure. Hence, under this circumstance, the pre-generated HIP associations can only be used to deliver revocation certificates, as it is difficult for the communicating partners to know whether the HI is still secure when the HIP associations were generated.

7.2. HI Refreshment

In key management mechanisms, key refreshment is concerned with the issues of using new cryptographic keys to take place of "old" ones. Therefore, it closely related with key revocation. A refreshment procedure of a key can occur either before or after the revocation of the key (Note that in the first case the key is still valid). In this section, we briefly discuss the issues with HI refreshment in HIP.

Ideally, the refreshment of an operational HI should be performed before its crypt-period is expired. That is, when an HI refreshment process is performed, the HI expected to be updated is still valid. The holder then can use the old HI to establish secure channels, and use Update packets to transport the refreshment information to related partners (in a push model) or to trusted third parties (in a pull model). In the Update packets, the new HI and other related information are encapsulated. Therefore, before the old HI expires, both HIs are valid, and the HIP associations generated with the old HI can still be applied.

In practice, the third parties deployed for HI revocation can also be used to support HI refreshment. For instance, when using a pull model, a host can transport the HI revoking and the refreshing information to a third party. Therefore, when a user inquires of the third party about the status information of an HI, the user can get the status of the HI inquired about as well as the associated refreshment information.

If an HI needs to be revoked due to accident disclosure or compromise, the update of the HI can be a little more complex. Although the invalid key can be used to send a "suicide" information to others (e.g., resolution systems, RVSes, or any entities which may be affected by the revocation), it cannot be used to securely transport the refreshment information any more.

If a host has multiple HIs, it can select a HI still valid to securely transport the refreshment information. The refreshment information should consist of both the new HI and the compromised HI. This solution requires that the partner communicating with the host can ensure that the HI used to generate secure channel and the compromised HI are possessed by the same HIP host. Such knowledge

can be obtained from resolution systems or provided by the host.

In the cases where all the HIs of a host become invalid (e.g., the host is found to compromised), the host only can distribute the refreshment information using an out-of-band way.

A host can also implement a pull model by directly transporting the update information to resolution servers. If the information is forwarded to a DNS server, users can query the latest HI using FQDN of the host. In a resolution system providing ID to locator mapping services (e.g., DHT), users can only try to query the resolution systems using old HITs. In this case, besides the IP addresses inquired, the resolution system should also provide the latest HIs and other useful information. Note that it is assumed that no two HITs of different hosts are identical, even if they are adopted in different period. In practice, because the length of HITs is long, the possibility that two hosts select a same HI can be very low. In order to further reduce the possibility, a user can also provide the life period of the inquired HIT in a query.

8. Conclusions

Key revocation is critical for HIP to be secure, practical and manageable. Particularly, HIP hosts are expected to keep working securely for a relatively long period, proper key revocation mechanisms for HIs must be provided. This document focuses on cons and pros of different key revocations and analyzes their security and practicality in different practical scenarios. Although key management has been an active research area for a long period and lots of successful key-management systems (e.g., PKI) are widely adopted in practice, many issues (e.g., scalability, lack of trust) still exist. There is no solution being found to meet the timeliness and performance requirements of all applications and environments that HIP is expected to support [McDaniel et al. 2001]. Therefore, it is predicted that various HI revocation approaches will be adopted after HIP has been globally adopted.

Because the HI of a HIP host acts as both the identity and the public key of the HIP host at the same time. The revocation of a HI, the identity of the host is changed. Without the assistance of other measures, the host will be regarded as a different one by others. For instance, during the revocation of a HI, all the TCP sessions identified with the assocatied HIT have to be broken.

The update of HIs is not rare, although it is relatively infrequent in compare with the change of IP addresses. The unstability issue introduced by the HI revocation must be considered in designing

identity management and resolution systems for HIP hosts. For instance,

9. IANA Considerations

This document makes no request of IANA.

10. Security Considerations

The whole document is about security.

11. Acknowledgements

Many Thanks to Thomas.R.Henderson for his kindly revision and precious comments.

12. References

12.1. Normative References

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

[RFC2459] Housley, R., Ford, W., Polk, T., and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and CRL Profile", RFC 2459, January 1999.

[RFC5201] Moskowitz, R., Nikander, P., Jokela, P., and T. Henderson, "Host Identity Protocol", RFC 5201, April 2008.

[RFC5204] Laganier, J. and L. Eggert, "Host Identity Protocol (HIP) Rendezvous Extension", RFC 5204, April 2008.

[RFC5205] Nikander, P. and J. Laganier, "Host Identity Protocol (HIP) Domain Name System (DNS) Extensions", RFC 5205, April 2008.

12.2. Informative References

[McDaniel et al. 2001]
McDaniel, P. and A. Rubin, "A Response to "can we eliminate certificate revocation list?"", 2001.

[Menezes et al. 1996]

MENEZES, A., VAN OORSCHOT, P., and S. AND VANSTONE, "Handbook in Applied Cryptography", 1996.

[Merwe et al. 2007]
Merwe, J., Dawoud, D., and S. McDONALD, "A Survey on Peer-to-Peer Key Management for Mobile Ad Hoc Networks", 2007.

[Recommendations]
Barker, E., Barker, W., Burr, W., Polk, W., and M. Smid, "Recommendation for Key Management-Part1-General(Revised)", March 2007.

Authors' Addresses

Dacheng Zhang
Huawei Technologies Co.,Ltd
HuaWei Building, No.3 Xinxi Rd., Shang-Di Information Industry Base, Hai-Dian District
Beijing, 100085
P. R. China

Phone:
Fax:
Email: zhangdacheng@huawei.com
URI:

Dmitriy Kuptsov
HIIT
Helsinki Institute for Information Technology
PO. Box 9800, TKK FI-02015
Finland

Phone:
Fax:
Email: dmitriy.kuptsov@hiit.fi
URI:

Sean Shen
CNNIC
4, South 4th Street, Zhongguancun
Beijing, 100190
P.R. China

Phone:
Fax:
Email: shenshuo@cnnic.cn
URI:

HIP Research Group
Internet Draft
Intended status: Experimental

Expires: September 2011

Pascal Urien
Telecom ParisTech
Gyu Myoung Lee
Telecom SudParis
Guy Pujolle
LIP6
March 2011

HIP support for RFIDs
draft-irtf-hiprg-rfid-02

Abstract

This document describes an architecture based on the Host Identity Protocol (HIP), for active RFIDs, i.e. Radio Frequency Identifiers inluding tamper resistant computing resources, as specified for example in the ISO 14443 or 15693 standards. HIP-RFIDs never expose their identity in clear text, but hide this value (typically an EPC-Code) by a particular equation (f) that can be only solved by a dedicated entity, referred as the portal. HIP exchanges occurred between HIP-RFIDs and portals; they are shuttled by IP packets, through the Internet cloud.

Requirements Language

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 [RFC2119].

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 2011.

Table of Contents

1 Overview

1.1 Motivation

RFIDs are electronic devices, associated to things or computers, who transmit their identity (usually a serial number) via radio links.

The first motivation for designing HIP support for RFIDs is to enforce a strong privacy for the Internet of Things, e.g. identity is protected by cryptographic procedures compatible with RFID computing resources. As an illustration EPC codes or IP addresses are today transmitted in clear form.

The second motivation is to define an identity layer for RFIDs logically independent from the transport facilities, which may optionally support IP stacks.

In other words we believe that the Internet of Things will be Identity oriented; RFIDs will act as electronic ID for objects to which they are linked. In this context privacy is a major challenge.

1.2 Passive and active RFIDs

An RFID is a slice of silicon whose area is about 1 mm2 for components used as cheap electronic RFIDs, and around 25 mm2 for chips like contact-less smart cards inserted in passports and mobile phones.

RFIDs are divided into two classes, the first includes devices that embed CPU and memories (RAM, ROM, E2PROM) such as contact-less smart cards, the second comprises electronic chips based on cabled logic circuits.

There are multiple standards relative to RFIDs.

The ISO 14443 standard introduces components dealing with the 13,56Mhz frequency that embed a CPU and consume about 10mW; data throughput is about 100 Kbits/s and the maximum working distance (from the reader) is around 10cm.

The ISO 15693 standard also uses the same 13,56 MHz frequency, but enables working distances as high as one meter, with a data throughput of a few Kbits/s.

The ISO 18000 standard defines parameters for air interface communications associated with frequency such as 135 KHz, 13.56 MHz, 2.45 GHz, 5.8 GHz, 860 to 960 MHz and 433 MHz. The ISO 18000-6 standard uses the 860-960 MHz range and is the basis for the Class-1 Generation-2 UHF RFID, introduced by the EPCglobal [EPCGLOBAL] consortium.

1.3 About the Internet of Things (IoT)

The term "Internet of Thing (IoT)" was invented by the MIT Auto-ID Center, in 2001, and refers to an architecture that comprises four levels,

- Passive RFIDs, such as Class-1 Generation-2 UHF RFIDs, introduced by the EPC Global consortium and operating in the 860-960 MHz range.

- Readers plugged to a local (computing) system, which read the Electronic Product Code [EPC].

- A local system, offering IP connectivity, which collects information pointed by the EPC thanks to a protocol called Object Naming Service (ONS)

- EPCIS (EPC Information Services) servers, which process incoming ONS requests and returns PML (Physical Markup Language) files [PML], e.g. XML documents that carry meaningful information linked to RFIDs.

1.4 HIP-RFIDs

This document suggests embedding a modified version of HIP stack in active RFIDs, named HIP-RFIDs. It assumes that such devices would not support an IP stack, but should be rather identity oriented, i.e. will use readers IP resources in order to unveil their EPC-Code only to trusted entities (called portals in the architecture shown by Figure 1). Privacy, e.g. identity protection seems a key prerequisite [SEC] before the effective massive deployment of these devices.

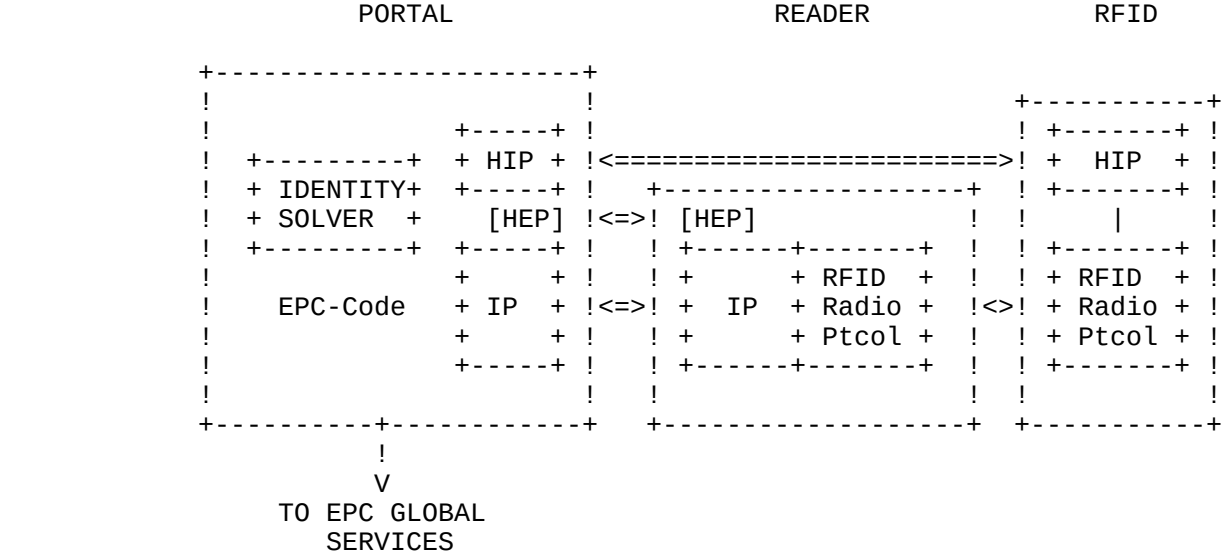


Figure 1. HIP-RFID Architecture

The functional HIP-RFID architecture includes three logical entities,

- HIP RFIDs. HIP is transported by IP packets. HIP-RFIDs support a modified version of this protocol but don’t require end-to-end IP transport.

- RFID readers. They provide IP connectivity and communicate with RFIDs through radio link either defined by EPC Global or ISO standards. The IP layer transports HIP messages between RFIDs and other HIP entities. According to HIP, an SPI (Security Parameter Index) associated to an IPSEC tunnel MAY be used by the IP host (e.g. a reader) in order to route HIP packets to/from the right software identity.

- HEP, HIP Encapsulation Protocol. HIP messages MAY be encapsulated by protocols such as UDP or TCP in order to facilitate HIP transport in existing software and networking architectures. The HEP does not modify the content of an HIP packet. This class of protocol is not specified by this document.

- PORTAL entity. This device manages a set of readers; it is an HIP entity that includes a full IP stack. Communications between portal and RFIDs logically work as peer to peer HIP exchanges. RFID identity (HIT) is hidden and appears as a pseudo random value; within the portal a software block called the IDENTITY SOLVER resolves an equation f, whose solution is an EPC Code. The portal accesses to EPCIS services; when required privacy may be enforced by legacy protocol such as SSL or IPSEC.

- The portal maintains a table linking HIT and EPC-Code. It acts as a router for that purpose it MUST provide an identity resolution mechanism, i.e. a relation between HIT and EPC-Code.

1.5 Main differences between HIP-RFID and HIP

In HIP [HIP], the HIT (Host Identifier Tag) is a fix value obtained from the hash of an RSA public key. This parameter is therefore linked to a unique identity, and can be used for traceability purposes; in other words HIP does not natively include privacy features.

In [BLIND], it is proposed to hide the HIT with by random number thanks to a hash function, i.e.

B-HIT = sha1(HIT || N), with N a random value and || the concatenation operation.

The case in which only one HIT (either initiator or responder) is blinded looks similar to the HIP-RFID protocol described in this draft working with a particular transform (HMAC Transform, 0x0001)

2. Basic Exchange

The HIP-RFID basic exchange (T-BEX) is derived from the "classical" BEX exchange, introduced in [HIP]. It is a four ways handshake illustrated by Figure 2.

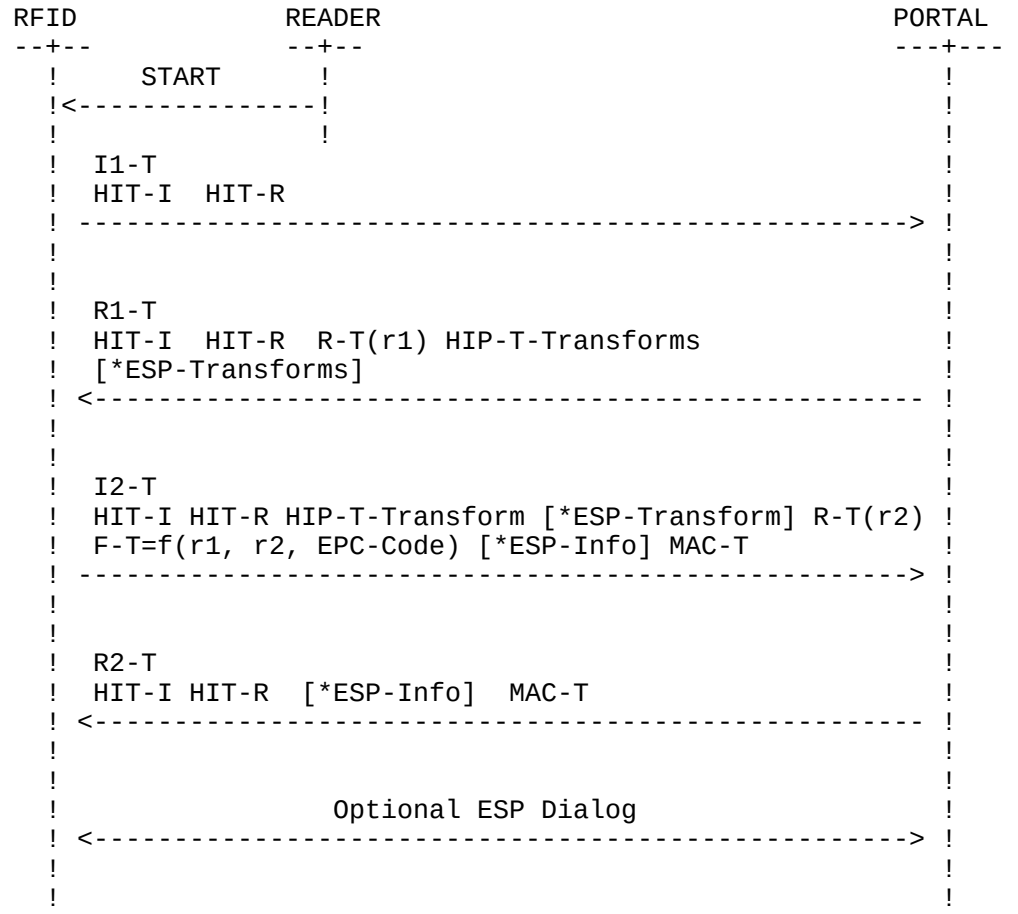


Figure 2. HIP-RFIDs Basic Exchange (T-BEX), *means optional attributes

A HEP layer MAY be used to transport HIP messages in non IP context, but this optional facility is out of scope from this document.

2.1 I1-T

When a reader detects an RFID, it realizes all low level operations in order to set up a radio communication link. Finally the reader delivers a START message that trigs the RFID.

The HIP-RFID sends the I1-T packet (I suffix meaning initiator), in which HIT-I is a true random value internally generated by the HIP-RFID.

If the RFID doesn’t known the portal HIT it sets the HIT-R value to zero; in that case the reader MAY modify this field in order to identify the appropriate entity.

The I1-T message is not MACed.

2.2 R1-T

The portal produces the R1-T (R suffix meaning responder) packet, which includes a nonce r1 and optional parameters. These fields indicate a list of supported authentication schemes (HIP-T-TRANSFORMs) and a list of ESP-TRANSFORMs, i.e. secure channels that could be opened between portal and RFIDs.

This message includes the following fields:
- HIT-I, a random number which identifies a RFID
- HIT-R, the portal HIP either a null or fix value.
- HIT-T-TRANSFORMs, a list of authentication schemes
- ESP-T-TRANSFORMs, an optional list of ESP secure channels

The R1-T message is not MACed.

2.3 I2-T

The HIP-RFID builds the I2-T message, which contains

- The selected HIP-T-TRANSFORM (the current authentication scheme).
- An optional ESP-TRANSFORM (a class of secure channel between RFID and portal).
- A nonce r2, included in the R-T attribute.
- An equation f(r1, r2, EPC-Code), whose solution, according to the selected HIP-T-TRANSFORM, unveils the EPC-Code value.
- An optional ESP-Info attribute that gives information about the secure (ESP) channel, and which includes the SPI-I value.
- A keyed MAC (MAC-T), which works with a KI-Auth-key deduced from r1, r2 and the hidden EPC-CODE value.

KI-Auth-key = g(r1, r2, EPC-Code)

The keyed MAC is computed over the complete I2-T message, the content of MAC-T resulting from this calculation is initially set to a nul value

The portal and the RFID shares secret keys. The meaning of these keys are dependent upon the f equation.

In some cases the EPC-Code is the only shared key. The portal knows a list of EPC-Code and tries all solutions for solving f, according to brute force techniques. As an illustration a hash function may be used for f:

f= sha1(r1 || r2 || EPC-Code), where || is the concatenation operation.

In other cases a set of keys is shared between portal and RFIDs. For example a binary tree of HMAC procedure MAY be used, each HMAC beeing associated to a particular key. A binary tree of depth n may identify 2**n RFIDs, each of them stores n keys (ki:j). The f function is a list of n values such as

HMAC(r1 || r2, ki:j)

Where ki:j is a secret key, and j the bit value (either 0 or 1) at the rank i (ranging between 0 and n-1)for the EPC-Code (or RFID index).

2.4 R2-T

The fourth and last R2-T packet is optional. It includes

- A keyed MAC (MAC-T) computed with the KI-Auth-key deduced from r1, r2 and the hidden EPC-CODE value.

KI-Auth-key = g(r1, r2, EPC-Code)

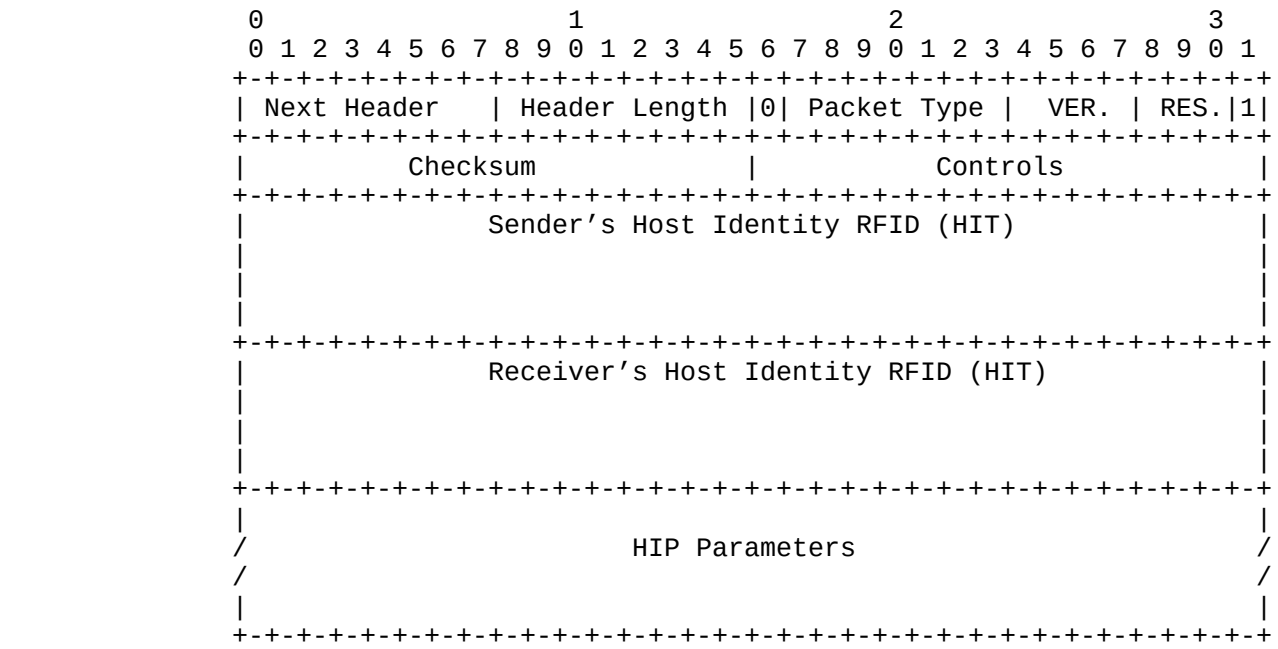
- An optional ESP-Info attribute that gives information about the secure (ESP) channel, and which includes the SPI-R value.

The R2-T packet is mandatory when an ESP channel has been previously negotiated. ESP channel is required if the portal intends to perform read or write operations with the RFIDs.

3. Formats

3.1 Payload

The payload format is imported from the [HIP] specification.



Next Header : normal value is decimal 59, IPPROTO_NONE.

Header Length: the length of the HIP Header and HIP parameters in 8 bytes units, excluding the first 8 bytes

Packet Type: Detailed in section 4.2

VER: 0001

RES: 000

Checksum: This checksum covers the source and destination addresses in the IP header.

HIP-RFIDs always deliver HIP packets with the null value for the checksum field. The reader MUST compute the checksum.

HIP-RFIDs do not check the checksum of received packets.

Controls: this field is reserved for future use (RFU)

Sender’s Host Identity RFID: 16 bytes HIT

Receiver’s Host Identity RFID: 16 bytes HIT

HIP Parameters: a list of attributes encoded in the TLV format

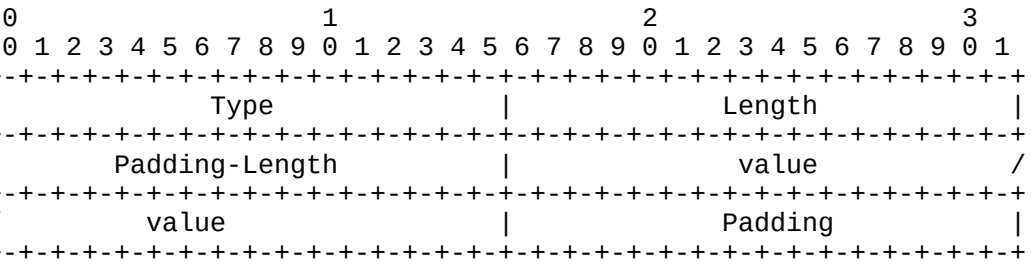
3.2 Packets types

Packet type	Packet name
0x40	I1-T - The HIP-RFID Initiator Packet
0x41	R1-T - The HIP-RFID Responder Packet
0x42	I2-T - The Second HIP-RFID Initiator Packet
0x43	R2-T - The Second HIP-RFID Responder Packet

3.3 Summary of HIP parameters

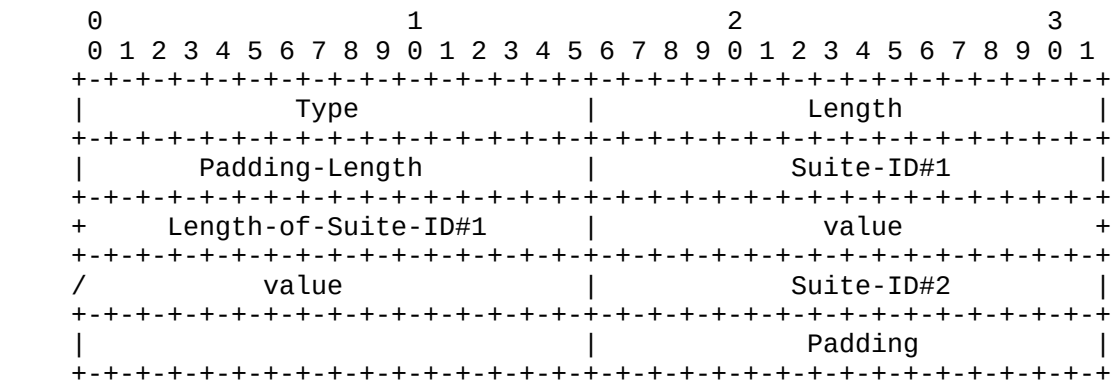
TLV	Type	Length	Data
R-T	0x400	variable	Random value r1 or r2
HIP-T-TRANSFORM	0x402	variable	HIP-RFID transform(s)
F-T	0x404	variable	f function value
MAC-T	0x406	variable	Keyed MAC
ESP-Transform	0x408	variable	ESP transform(s)
ESP-Info	0x40A	variable	ESP parameter(s)

3.4 R-T



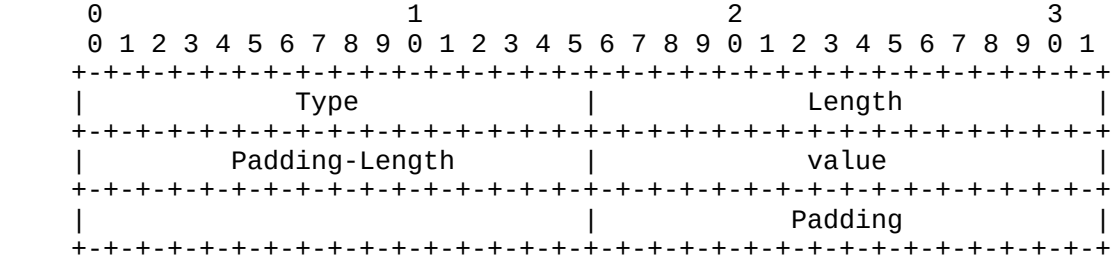
Type	0x400
Length	total length in bytes
Value	random value
Padding-Length	padding length in bytes
Padding	padding bytes

3.5 HIP-T-Transform



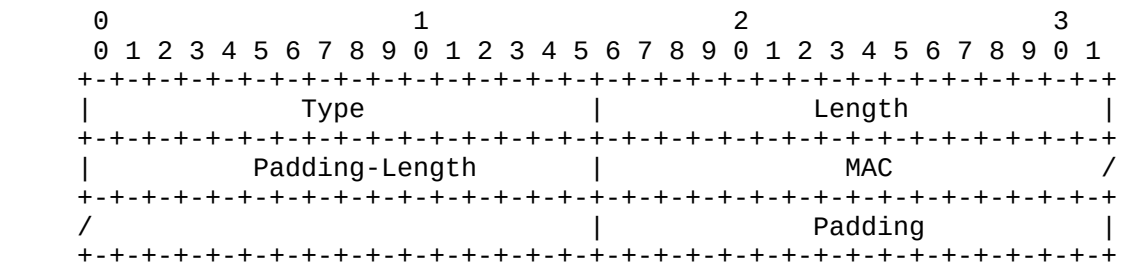
Type	0x402
Length	Total length
Padding-Length	Number of padding bytes
Suite-ID	Defines the HIP Cipher Suite to be used
Length-of-Suite-ID	Defines the length of optional data
Padding	Padding bytes

3.6 F-T



Type	0x404
Length	total length, in bytes
Padding-Length	padding length in bytes
Value	f value
Padding	padding bytes

3.7 MAC-T



Type	0x406
Length	total length, in bytes
Padding-Length	padding length, in bytes
Value	Keyed MAC value
Padding	padding bytes

A MAC procedure works with the K-Auth-Key and is computed over the whole HIP message according to the following rules

- The checksum field of the HIP header is set to a null value.

- The MAC field of the MAC-T attribute is set to a null value

3.8 ESP-Transform

Details of the attribute will be specified by another document.

3.9 ESP-Info

Details of the attribute will be specified by another document.

4. BEX Example

4.1 Generic example

4.1.1 I1-T

Next Header:	0x3B
Header Length:	0x4
Packet Type:	0x40
Version:	0x1
Reserved:	0x1
Control:	0x0
Checksum:	0x0000
Sender's HIT (RFID) :	0x0123456789ABCDEF 0123456789ABCDEF
Receiver's HIT (Portal) :	0x0000000000000000 0000000000000000

The checksum is computed by portal and reader according to rules specified in [HIP]; it covers the source and destination IP addresses.

4.1.2 R1-T

Next Header:	0x3B
Header Length:	0xB
Packet Type:	0x41
Version:	0x1
Reserved:	0x1
Control:	0x0
Checksum:	0xabcd
Sender's HIT (Portal)	0xA5A5A5A5A5A5A5A5 5A5A5A5A5A5A5A5A
Receiver's HIT (RFID)	0x0123456789ABCDEF 0123456789ABCDEF
R-T	0x040000280002rrrr rrrrrrrrrrrrrrrr rrrrrrrrrrrrrrrr rrrrrrrrrrrrrrrr rrrrrrrrrrrrpppp
HIP-T-Transforms	0x0402001000020001 000000020000pppp

r1 is a 128 bits value
Transforms 1, 2 are supported by the reader.

4.1.3 I2-T

Next Header:	0x3B
Header Length:	0x14
Packet Type:	0x42
Version:	0x1
Reserved:	0x1
Control:	0x0
Checksum:	0x0000
Sender's HIT (RFID) :	0x0123456789ABCDEF 0123456789ABCDEF
Sender's HIT (Portal) :	0xA5A5A5A5A5A5A5A5 5A5A5A5A5A5A5A5A
HIP-T-Transform	0x0402001000060001 0000pppppppppppp
R-T	0x040000280002rrrr rrrrrrrrrrrrrrrr rrrrrrrrrrrrrrrr rrrrrrrrrrrrrrrr rrrrrrrrrrrrpppp
F-T	0x040400280002ffff ffffffffffffffff ffffffffffffffff ffffffffffffffff fffffffffffpppp
MAC-T	0x040600040006ssss ssssssssssssssss ssssssssssssssss sssspppppppppppp

The RFID selects the HIP-Transform number one. It produces an r2 nonce and computes a f value. It appends a 20 bytes keyed MAC.

4.1.4 R2-T

Next Header:	0x3B
Header Length:	0x08
Packet Type:	0x40
Version:	0x1
Reserved:	0x1
Control:	0x0
Checksum:	0xabcd
Sender's HIT (RFID) :	0x0123456789ABCDEF 0123456789ABCDEF
Sender's HIT (Portal) :	0xA5A5A5A5A5A5A5A5 5A5A5A5A5A5A5A5A
MAC-T	0x040600040006ssss ssssssssssssssss ssssssssssssssss sssspppppppppppp

Reader ends the BEX-T.

4.2 HIP-T Transform 0x0001, HMAC

EPC = 0123456789abcdefcdab

4.2.1 I1-T

```
<< 3B 04 40 11 00 00 00 00 6A 68 2E 53 51 6B 51 6F
   2F 58 CE 60 25 42 1A E6 00 00 00 00 00 00 00 00
   00 00 00 00 00 00 00 00

HEAD 3b04401100000000
sHIT 6a682e53516b516f2f58ce6025421ae6
dHIT 00000000000000000000000000000000
```

4.2.2 R1-T

```
>> 3B 0A 41 11 00 00 00 00 00 00 00 00 00 00 00 00
   00 00 00 00 00 00 00 00 6A 68 2E 53 51 6B 51 6F
   2F 58 CE 60 25 42 1A E6 04 00 00 20 00 06 27 6D
   03 4D DD 2D 52 79 3B 17 2C B9 5B CD 02 97 E2 DF
   61 15 00 00 00 00 00 00 04 02 00 10 00 06 00 02
   00 00 00 00 00 00 00 00

HEAD 3b0a411100000000
sHIT 00000000000000000000000000000000
dHIT 6a682e53516b516f2f58ce6025421ae6

ATT 0400 20 bytes  276d034ddd2d52793b172cb95bcd0297e2df6115
ATT 0402 04 bytes  00020000
```

4.2.3 I2-T

```
<< 3B 13 40 11 00 00 00 00 6A 68 2E 53 51 6B 51 6F
   2F 58 CE 60 25 42 1A E6 00 00 00 00 00 00 00 00
   00 00 00 00 00 00 00 00 04 02 00 10 00 06 00 01
   00 00 00 00 00 00 00 00 04 00 00 20 00 06 C5 95
   8B 23 6B 9B 0E AA 7A BB 25 F2 7D 24 C5 04 6E 89
   19 9E 00 00 00 00 00 00 04 04 00 20 00 06 80 1D
   BC 55 C5 F3 97 89 F8 3C 6C BA 14 50 18 7D 83 83
   3C AF 00 00 00 00 00 00 04 06 00 20 00 06 2A 23
   68 93 2B F7 3A BE C4 6B DD B8 3F 1B 3F 7F 9D ED
   8B 83 00 00 00 00 00 00

HEAD 3b13401100000000
sHIT 6a682e53516b516f2f58ce6025421ae6
dHIT 00000000000000000000000000000000

ATT 0402 04 bytes  00010000
ATT 0400 20 bytes  c5958b236b9b0eaa7abb25f27d24c5046e89199e
ATT 0404 20 bytes  801dbc55c5f39789f83c6cba1450187d83833caf
ATT 0406 20 bytes  2a2368932bf73abec46bddb83f1b3f7f9ded8b83
```

5. HIP-T-Transforms Definition

5.1 Type 0x0001, HMAC

5.1.1 Suite-ID

```
Suite-ID:           0x0001
Length-of-Suite-ID: 0x0000
```

5.1.2 F-T computing (f function)

The F-T function produces a 20 bytes result, according to the relation:

K = HMAC-SHA1(r1 | r2, EPC-Code)

Y = f(r1, r2, EPC-Code) = HMAC-SHA1(K, CT1 | "Type 0001 key")

Where:

- SHA1 is the SHA1 digest function

- EPC-Code is the RFID identity

- HMAC-SHA1 is the keyed MAC algorithm based on the SHA1 digest procedure.

- CT1 is a 32 bits string, whose value is equal to 0x00000001

- r1 and r2 are the two random values exchanged by the BEX

5.1.3 K-Auth-Key computing (g function)

The K-Auth-Key is computing according to the relation:

K = HMAC-SHA1(r1 | r2, EPC-Code)

Y = HMAC-SHA1(K, CT2 | "Type 0001 key")

Where:

- SHA1 is the SHA1 digest function

- EPC-Code is the RFID identity

- HMAC-SHA1 is the keyed MAC algorithm based on the SHA1 digest procedure.

- CT2 is a 32 bits string, whose value is equal to 0x00000002

- r1 and r2 are the two random values exchanged by the BEX

5.1.4 MAC-T computing

The HMAC-SHA1 function is used with the K-Auth-Key secret value:

MAC-T(HIT-T packet) = HMAC-SHA1(K-Auth-Key, HIP-T packet)

5.2 Type 0x0002, Keys-Tree

5.2.1 Suite-ID

Suite-ID: 0x0002
Length-of-Suite-ID: 0x0006
Value1: an index identifying a HASH function (H), which produces t bytes.
Value2: n, the depth of the tree, a two bytes number.
Value3: p, the maximum number of child nodes, for each node, a two bytes number.

The maximum elements of a keys-tree is therefore p**n

5.2.2 F-T computing (f function)

The F-T function produces a list of Hi, 1<= i <= n, of nh bytes results, according to the relation:

Y = f(r1, r2, EPC-Code) = H1 | H2 | Hi | Hn

With
Hi = HMAC-SHA1(r1 | r2, Ki:j)

Where:

- H is digest function producing t bytes

- Ki:j is a set of pn secret keys.

Each EPC-Code is associated with an index, whose value is written as:

RFID-Index = an p**(n-1) + an-1 p**(n-2) + a1

Each ai digit(ai p**(i-1))whose value ranges between 0 and p-1, is associated with a key Ki:j (i.e. the tree is made with pn keys, but only n values are stored in a given RFID), with j=ai

- HMAC-H is the keyed MAC algorithm based on the H digest procedure.

- r1 and r2 are the two random values exchanged by the BEX.

5.2.3 K-Auth-Key computing (g function)

The K-Auth-Key is computing according to the relation:

K-Auth-Key = HMAC-H(r1 | r2, RFID-Index)

Where:

- H is a digest function producing t bytes

- HMAC-H is the keyed MAC algorithm based on the H digest procedure.

- RFID INDEX is the RFID index.

- r1 and r2 are the two random values exchanged by the BEX.

5.2.4 MAC-T computing

The HMAC-H function is used with the K-Auth-Key secret value:

MAC-T(HIT-T packet) = HMAC-H(K-Auth-Key, HIP-T packet)

6. Security Considerations

To be done.

7. IANA Considerations

To be done.

8 References

8.1 Normative references

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

[HIP] R. Moskowitz, P. Nikander, P. Jokela, T. Henderson, Host Identity Protocol, RFC 5201, April 2008.

8.2 Informative references

[EPC] Brock, D.L, The Electronic Product Code (EPC), A Naming Scheme for Physical Objects, MIT AUTO-ID CENTER, 2001.

[PML] Brock, D.L - The Physical Markup Language, MIT AUTO-ID CENTER, 2001.

[EPCGLOBAL] EPCglobal, EPC Radio Frequency Identity Protocols Class 1 1516 Generation 2 UHF RFID Protocol for Communications at 860 MHz-960 MHz Version 1517 1.0.9, EPCglobal Standard, January 2005.

[NIST-800-108] NIST Special Publication 800-108, Recommendation for Key Derivation Using Pseudorandom Functions.

[SEC] S. Weis, S. Sarma, R. Rivest and D. Engels. "Security and privacy aspects of low-cost radio frequency identification systems" In D. Hutter, G. Muller, W. Stephan and M. Ullman, editors, International Conference on Security in Pervasive Computing - SPC 2003, volume 2802 of Lecture Notes in computer Science, pages 454-469. Springer-Verlag, 2003.

[HIP-TAG-EXP] Pascal Urien, Simon Elrharbi, Dorice Nyamy, Herve Chabanne, Thomas Icart, Francois Lecocq, Cyrille Pepin, Khalifa Toumi, Mathieu Bouet, Guy Pujolle, Patrice Krzanik, Jean-Ferdinand Susini, "HIP-Tags architecture implementation for the Internet of Things", AH-ICI 2009. First Asian Himalayas International Conference on Internet, 3-5 Nov. 2009.

[BLIND] Dacheng Zhang, Miika Komu, "An Extension of HIP Base Exchange to Support Identity Privacy", draft-zhang-hip-privacy-protection-00, work in progress, March 2010.

9 Annex I

This annex provides a sample code, for NFC RFIDs working at 13.56 Mhz and implementing a Java Virtual Machine.

9.1 Binary Interface with HIP RFIDs

According to the ISO 7816 standards, embedded RFID applications are identified by an AID attribute (Application IDentifier) whose size ranges between 5 and 16 bytes.

Commands exchanged between RFIDs and readers are named APDUs and are associated with a short prefix, whose size is usually 5 bytes referred as CLA, INS, P1, P2, P3.

In our sample we choose an arbitrary value for the AID (11223344556601, in hexadecimal representation) and a unique command CLA=00, INS=C2, P1=00, P2=00. The P3 byte is set to null in order to trig the RFID (which resets its state machine and returns the I1 packet, or a non null value when it pushes the R1 packet.

9.3 Exchanged data

The reader selects the embedded HIP-RFID application.

```
>> 00 A4 04 00 07 11 22 33 44 55 66 01
<< 90 00
```

The reader trigs the first packet I1-T.

```
>> 00 C2 00 00 00
```

The RFID delivers the I1-T packet.

```
<< 3B 04 40 11 00 00 00 00 A3 12 9D 5E 28 16 67 4F FC 4F A8 08 4E 30
55 E8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 90 00
```

The reader forwards the R1-T packet to the HIP RFID.

```
>> 00 C2 00 00 58 3B 0A 41 11 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 A3 12 9D 5E 28 16 67 4F FC 4F A8 08 4E 30 55 E8
04 00 00 20 00 06 68 46 95 15 02 10 32 C2 B7 8D 13 E7 53 F6 25 0F 09
AD 7A BD 00 00 00 00 00 00 04 02 00 10 00 06 00 01 00 00 00 00 00 00
00 00
```

The RFID produces the I2-T packet.

```
<< 3B 13 40 11 00 00 00 00 A3 12 9D 5E 28 16 67 4F FC 4F A8 08 4E 30
55 E8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 04 02 00 10 00
06 00 01 00 00 00 00 00 00 00 00 04 00 00 20 00 06 71 3A DD 19 C4 CB
59 D4 AF D0 2B FD F9 7C 2F 8A D1 23 32 E0 00 00 00 00 00 00 04 04 00
20 00 06 70 DA C1 F7 0B CA 63 15 57 CB D7 AA 66 A9 FD 36 B4 1F DB E3
00 00 00 00 00 00 04 06 00 20 00 06 A6 A7 00 67 5D FD A9 2F 3E 5C 00
D6 B0 8A 55 A2 99 D8 86 79 00 00 00 00 00 00 90 00
```

9.3 Javacard code sample

```
package hiprfid;

// Author Pascal Urien

import  javacard.framework.*;
import  javacard.security.* ;


public class rfid extends Applet
{
 final static byte  SELECT          = (byte)0xA4 ;
 final static byte  INS-HIP         = (byte)0xC2 ;

 final static short R-T             = (short)0x400 ;
 final static short HIP-T-TRANSFORM = (short)0x402 ;
 final static short F-T             = (short)0x404 ;
 final static short Signature-T     = (short)0x406 ;
 final static short ESP-Transform   = (short)0x408 ;
 final static short ESP-Info        = (short)0x40A ;

 final static int ALIGN = 8;
 final static short len-r2 =(short)20;
 final byte[] algo1 = {(byte)0x00,(byte)0x01,(byte)0x00,(byte)0x00 };

 final byte[] ct1 =  {
 (byte)0x00,(byte)0x00,(byte)0x00,(byte)0x01,
 (byte)'T',(byte)'y', (byte)'p',(byte)'e',
 (byte)' ',(byte)'0',(byte)'0',(byte)'0',(byte)'1',
 (byte)' ',(byte)'k',(byte)'e',(byte)'y' };

 final byte[] ct2 =  {
 (byte)0x00,(byte)0x00,(byte)0x00,(byte)0x02,
 (byte)'T',(byte)'y',(byte)'p',(byte)'e',
 (byte)' ',(byte)'0',(byte)'0',(byte)'0',(byte)'1',
 (byte)' ',(byte)'k',(byte)'e',(byte)'y' };

 MessageDigest sha1=null ;
 RandomData rnd=null;
 byte[] DB =null;
 final static short DBSIZE=(short)200;
 final static short off-myHIT = (short)0   ;
 final static short off-rHIT =  (short)16  ;
 final static short off-R1   =  (short)32  ;
 final static short off-R2   =  (short)64  ;
 final static short off-kaut =  (short)96  ;
 final static short off-k    =  (short)128 ;
 final static short off-FT   =  (short)160 ;
```

```
final byte[] HEADER= {
 (byte)0x3b,(byte)0x04,(byte)0x40,(byte)0x11,
 (byte)0x00,(byte)0x00,(byte)0x00,(byte)0x00 };


 final byte[] MyEPCCODE = {
 (byte)0x01,(byte)0x23,(byte)0x45,(byte)0x67,(byte)0x89,
 (byte)0xab,(byte)0xcd,(byte)0xef,(byte)0xcd,(byte)0xab };

 public void init(){
 try { sha1=MessageDigest.getInstance(MessageDigest.ALG-SHA,false);}
 catch (CryptoException e){sha1=null;}

 try { rnd = RandomData.getInstance(RandomData.ALG-SECURE-RANDOM);}
 catch (CryptoException e){}

 DB = JCSystem.makeTransientByteArray(DBSIZE,
                                      JCSystem.CLEAR-ON-DESELECT);

}

public short GetAttOffset(byte[] pkt, short off, short len,short att)
{ boolean more=true;
  short type=(short)0;
  short tl=(short)0;

  if (len <= (short)40) return (short)-1 ;

  while (more)
  { type = Util.getShort(pkt,off)           ;
    tl   = Util.getShort(pkt,(short)(off+2));
    if (type == att) return off    ;
    off =(short)(off+tl) ;
    if (off >= (short)(off+len))more=false;
  }

 return -1;
}


public static short GetPadLength(short size)
{
 if ( (short)(size % ALIGN) == (short)0) return (short)0;
 return (short)(ALIGN - size % ALIGN );
 }


public static short Set_Att(short att, byte[] ref-att, short off-att,
                            short len-att, byte[] pkt, short off)
{
  short tl = (short) (len-att + 6) ;
```

```
  short tp = GetPadLength(tl)      ;

  tl= (short) (tp+tl);

  Util.setShort(pkt,off,att)  ;
  Util.setShort(pkt,(short)(off+2),tl);
  Util.setShort(pkt,(short)(off+4),tp);

  if (ref_att != null)
  Util.arrayCopy(ref-att,off-att,pkt,(short)(off+6),len-att);
  else
  Util.arrayFillNonAtomic(pkt,(short)(off+6),len-att,(byte)0);

  if (tp != (short)0)
  Util.arrayFillNonAtomic(pkt,(short)(off+6+len-att),tp,(byte)0);

  return tl ;
 }


public void process(APDU apdu) throws ISOException
{
 short len=(short)0, readCount=(short)0;
 short off=(short)0,pad=(short)0,len-r1=(short)0;
 short size=(short)0;

 byte[] buffer = apdu.getBuffer() ; // CLA INS P1 P2 P3

 byte cla = buffer[ISO7816.OFFSET_CLA];
 byte ins = buffer[ISO7816.OFFSET_INS];
 byte P1  = buffer[ISO7816.OFFSET_P1] ;
 byte P2  = buffer[ISO7816.OFFSET_P2] ;
 byte P3  = buffer[ISO7816.OFFSET_LC] ;

 switch (ins)
 {
  case SELECT:
  size = apdu.setIncomingAndReceive();
  return;

 case INS_HIP:

  if (P3 == (byte)0)
  {
   rnd.generateData(DB,off_myHIT,(short)16);
   Util.arrayCopy(HEADER,(short)0,buffer,(short)0,(short)8);
   Util.arrayCopy(DB,off-myHIT,buffer,(short)8,(short)16)  ;
   Util.arrayFillNonAtomic(DB,(short)24,(short)16,(byte)0) ;
   apdu.setOutgoingAndSend((short)0,(short)40)             ;
   break;
  }
```

```
  else
  {
  size = apdu.setIncomingAndReceive();
  len  = Util.makeShort((byte)0,buffer[6]);
  len  = (short)(len << 3);
  len  = (short)(len+(short)8)   ;

  if (len != size) ISOException.throwIt(ISO7816.SW-DATA-INVALID) ;
  size = (short)(len-(short)40);

  // HEADER 00...08
  // HIT-S  08...24
  // HIT-D  24...40

  Util.arrayCopy(buffer,(short)13,DB,off_rHIT,(short)16);
  off= GetAttOffset(buffer,(short)45,size,R-T);
  if (off==(short)-1) ISOException.throwIt(ISO7816.SW-DATA-INVALID) ;
  len = Util.getShort(buffer,(short)(off+2));
  pad = Util.getShort(buffer,(short)(off+4));
  len = (short)(len-pad-6);

  len-r1=len;
  Util.arrayCopy(buffer,(short)(off+6),DB,off-R1,len);
  off= GetAttOffset(buffer,(short)45,size,HIP-T-TRANSFORM)      ;

  if (off==(short)-1) ISOException.throwIt(ISO7816.SW-DATA-INVALID) ;
  len = Util.getShort(buffer,(short)(off+2));
  pad = Util.getShort(buffer,(short)(off+4));
  len = (short)(len-pad-6);

  // algo=Util.getShort(buffer,(short)(off+6)
  rnd.generateData(DB,(short)(off-R1+len-r1),len-r2); // r1 || r2

  Util.arrayCopy(MyEPCCODE,(short)0,buffer,
                           (short)0,(short)MyEPCCODE.length);

  hmac(DB,off_R1,(short)(len-r1 + len-r2),
       buffer,(short)0,(short)MyEPCCODE.length,
       sha1,
       DB,off-k);

  Util.arrayCopy(ct1,(short)0,buffer,(short)0,(short)ct1.length);

  hmac(DB,off_k,(short)20,
       buffer,(short)0,(short)ct1.length,
       sha1,
       DB, off-FT);

  Util.arrayCopy(ct2,(short)0,buffer,(short)0,(short)ct2.length);
```

```
  hmac(DB,off-k,(short)20,
       buffer,(short)0,(short)ct2.length,
       sha1,
       DB, off-kaut);

  Util.arrayCopy(HEADER,(short)0,buffer,
                 (short)0,(short)HEADER.length);

  Util.arrayCopy(DB,off-myHIT, buffer, (short)8,(short)16);
  Util.arrayCopy(DB, off-rHIT, buffer,(short)24,(short)16);

  off=(short)40;
  len = Set-Att(HIP-T-TRANSFORM,algo1,
                 (short)0,(short)algo1.length,buffer,off);
  off = (short)(off+len);
  len = Set-Att(R-T,DB,(short)(off-R1+len-r1),len-r2,buffer,off);
  off = (short)(off+len);
  len = Set-Att(F-T,DB,off-FT,(short)20,buffer,off);
  off = (short)(off+len);
  len = Set-Att(Signature-T,null,(short)0,(short)20,buffer,off);
  size= (short)(off+len);
  buffer[1] = (byte) (size >>3);

 hmac(DB,off-kaut,(short)20,
      buffer,(short)0,size,
      sha1,
      buffer,(short)(off+6));

apdu.setOutgoingAndSend((short)0,size);
  break;
}

 default:
 ISOException.throwIt(ISO7816.SW-INS-NOT-SUPPORTED);
  }

}

protected rfid(byte[] bArray,short bOffset,byte bLength)
{init();
 register();
}

public static void install( byte[] bArray, short bOffset, byte
bLength )
{
new rfid(bArray,bOffset,bLength);
}
```

```
public boolean select()
{
return true;
}

public void deselect()
{
}
```

Author's Addresses

Pascal Urien
Telecom ParisTech
23 avenue d'italie, 75013 Paris, France

Email: Pascal.Urien@telecom-paristech.fr

Gyu Myoung Lee
Telecom SudParis
9 rue Charles Fourier, 91011 Evry, France

Email: gm.lee@it-sudparis.eu

Guy Pujolle
Laboratoire d'informatique de Paris 6 (LIP6)
4 place Jussieu
75005 Paris France

Email: Guy.Pujolle@lip6.fr

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 15, 2011

R. Moskowitz
Verizon

March 14, 2011

HIP Diet EXchange (DEX)
draft-moskowitz-hip-rg-dex-05

Abstract

This document specifies the details of the Host Identity Protocol Diet EXchange (HIP DEX). HIP DEX is a variant of the HIP Base EXchange (HIP BEX) [RFC5201-bis] specifically designed to use as few crypto primitives as possible yet still deliver the same class of security features as HIP BEX.

The design goal of HIP DEX is to be usable by sensor devices that are memory and processor constrained. Like HIP BEX it is expected to be used together with another suitable security protocol, such as the Encapsulated Security Payload (ESP). HIP DEX can also be used directly as a keying mechanism for a MAC layer security protocol as is supported by IEEE 802.15.4 [IEEE.802-15-4.2006].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 15, 2011.

Copyright Notice

Table of Contents

1. Introduction

This memo specifies the details of the Host Identity Protocol Diet EXchange (HIP DEX). HIP DEX uses the smallest possible set of established cryptographic primitives, in such a manner that does not change our understanding of their behaviour, yet in a different formulation to achieve assertions normally met with different primitives.

HIP DEX builds on HIP BEX [RFC5201-bis], and only the differences between BEX and DEX are documented here.

There are a few key differences between BEX and DEX.

- Minimum collection of cryptographic primitives.
  - AES-CBC for symmetric encryption and to provide CMAC for MACing functions.
  - Static Elliptic Curve Diffie-Hellman key pairs used to encrypt the session key.
  - A simple truncation function for HIT generation.
- Forfeit of Perfect Forward Secrecy with the dropping of ephemeral Diffie-Hellman.
- Forfeit of digital signatures with the removal of a hash function. Reliance of DH derived key used in HIP_MAC to prove ownership of the private key.
- Provide a Password Authentication within the exchange. This may be supported by BEX as well, but not defined there.
- Operate in an aggressive retransmission manner to deal with the high packet loss nature of sensor networks.

1.1. The HIP Diet EXchange (DEX)

The HIP diet exchange is a two-party cryptographic protocol used to establish communications context between hosts. The first party is called the Initiator and the second party the Responder. The four-packet design helps to make HIP DoS resilient. The protocol exchanges Static Diffie-Hellman keys in the 2nd and 3rd packets, transmits session secrets in the 3rd and 4th packets, and authenticates the parties also in the 3rd and 4th packets. Additionally, the Responder starts a puzzle exchange in the 2nd packet, with the Initiator completing it in the 3rd packet before the

Responder stores any state from the exchange.

Thus DEX is operationally similar to BEX. The model is fairly equivalent to 802.11-2007 [IEEE.802-11.2007] Master Key and Pair-wise Transient Key, but handled in a single exchange.

HIP DEX does not have the option of encrypting the Host Identity of the Initiator in the 3rd packet. The Responder's Host Identity is also not protected. Thus there is no attempt at anonymity as in BEX.

Data packets start to flow after the 4th packet. Simiarly to HIP BEX, DEX does not have an explicit transition to connected state for the Responder.

This is learned when the Responder starts receiving protected datagrams, indicating that the Initiator received the R2 packet. As such the Intitator should take care to NOT send the first data packet until some delta time after it received the R2 packet. This is to provide time for the Responder to process any aggressively retransmitted I2 packets.

An existing HIP association can be updated using the update mechanism defined in this document, and when the association is no longer needed, it can be closed using the defined closing mechanism.

Finally, HIP is designed as an end-to-end authentication and key establishment protocol, to be used with Encapsulated Security Payload (ESP) [rfc5202-bis] and other end-to-end security protocols. The base protocol does not cover all the fine-grained policy control found in Internet Key Exchange (IKE) [RFC4306] that allows IKE to support complex gateway policies. Thus, HIP is not a replacement for IKE.

1.2. Memo Structure

The rest of this memo is structured as follows. Section 2 defines the central keywords, notation, and terms used throughout the rest of the document. Section 4 gives an overview of the HIP base exchange protocol. Section 6 define the rules for packet processing. Finally, Sections 7, 8, and 9 discuss policy, security, and IANA considerations, respectively.

2. Terms and Definitions

2.1. Requirements 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 RFC 2119 [RFC2119].

2.2. Notation

[x] indicates that x is optional.

{x} indicates that x is encrypted.

X(y) indicates that y is a parameter of X.

<x>i indicates that x exists i times.

--> signifies "Initiator to Responder" communication (requests).

<-- signifies "Responder to Initiator" communication (replies).

| signifies concatenation of information-- e.g., X | Y is the concatenation of X with Y.

Ltrunc (M(x), K) denotes the lowest order K bits of the result of the mac function M on the input x.

3. The DEX Host Identifier Tag (HIT) and Its Representations

The DEX Host Identity Tag (HIT) is distinguished in two ways from the BEX HIT:

The HIT SUITE ID Section 5.1.1 is ONLY a DEX ID.

The HIT DEX HIT is not generated via a cryptographic hash. Rather it is a truncation of the Elliptic Curve Host Identity.

3.1. Host Identity Tag (HIT)

The DEX Host Identity Tag is a 128-bit value -- a truncation of the Host Identifier appended with a prefix. There are two advantages of using a Host Identity Tag over the actual Host Identity public key in protocols. Firstly, its fixed length makes for easier protocol coding and also better manages the packet size cost of this technology. Secondly, it presents a consistent format to the protocol whatever underlying identity technology is used.

BEX uses RFC 4843-bis [RFC4843-bis] specified 128-bit hash-based identifiers, called Overlay Routable Cryptographic Hash Identifiers (ORCHIDs). Their prefix, allocated from the IPv6 address block, is defined in [RFC4843-bis].

In DEX, a cryptographic hash is NOT used to form the HIT. Rather the

HI is truncated to 96 bits.

3.2. Generating a HIT from an HI

The DEX HIT is not an ORCHID, as there is no hash function in DEX. Since a HI that is an ECDH key is directly computed from a random number it is already collision resistant. The DEX HIT is the left-truncated 96 bits of the HI. This 96 bit value is used in place of the hash in the ORCHID. The HIT suite (see Section 9) is used for the four bits of the Orchid Generation Algorithm (OGA) field in the ORCHID. The same IPv6 prefix used in BEX is used for DEX.

4. Protocol Overview

The following material is an overview of the differences between the BEX and DEX implementations of the HIP protocol. It is expected that [RFC5201-bis] is well understood first.

4.1. Creating a HIP Association

By definition, the system initiating a HIP exchange is the Initiator, and the peer is the Responder. This distinction is forgotten once the base exchange completes, and either party can become the Initiator in future communications.

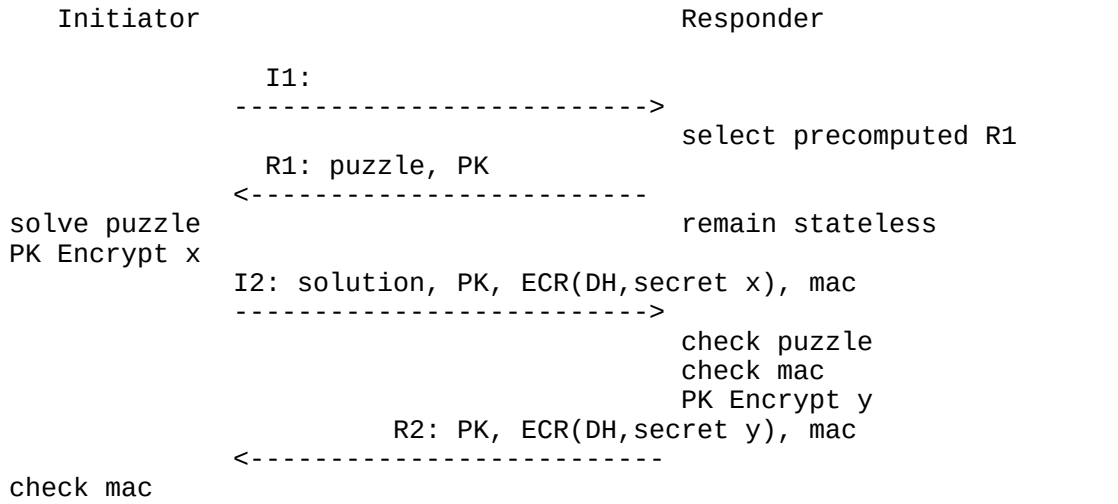
The HIP Diet EXchange serves to manage the establishment of state between an Initiator and a Responder. The first packet, I1, initiates the exchange, and the last three packets, R1, I2, and R2, constitute an authenticated secret key wrapped by a Diffie-Hellman derived key for session key generation. The HIP association keys are drawn from this keying material. If other cryptographic keys are needed, e.g., to be used with ESP, they are expected to be drawn from the same keying material.

The second packet, R1, starts the actual exchange. It contains a puzzle -- a cryptographic challenge that the Initiator must solve before continuing the exchange. The level of difficulty of the puzzle can be adjusted based on level of trust with the Initiator, current load, or other factors. The R1 also contains lists of cryptographic algorithms supported by the Responder. Based on these lists, the Initiator can continue, abort, or restart the base exchange with a different selection of cryptographic algorithms.

In the I2 packet, the Initiator must display the solution to the received puzzle. Without a correct solution, the I2 message is discarded. The I2 also contains a key wrap parameter that carries the key for the Responder. This key is only half the final session key. The packet is authenticated by the sender (Initiator).

The R2 packet finalizes the base exchange. The R2 contains a key wrap parameter that carries the rest of the key for the Initiator. The packet is authenticated by the sender (Initiator).

The base exchange is illustrated below. The term "key" refers to the Host Identity public key, "secret" refers to a random value encrypted by a public key, and "sig" represents a signature using such a key. The packets contain other parameters not shown in this figure.



4.1.1. HIP Puzzle Mechanism

The purpose of the HIP puzzle mechanism is to protect the Responder from a number of denial-of-service threats. It allows the Responder to delay state creation until receiving I2. Furthermore, the puzzle allows the Responder to use a fairly cheap calculation to check that the Initiator is "sincere" in the sense that it has churned CPU cycles in solving the puzzle.

DEX uses the CMAC function instead of a hash function as in BEX.

The puzzle mechanism has been explicitly designed to give space for various implementation options. It allows a Responder implementation to completely delay session-specific state creation until a valid I2 is received. In such a case, a correctly formatted I2 can be rejected only once the Responder has checked its validity by computing one CMAC function. On the other hand, the design also allows a Responder implementation to keep state about received I1s, and match the received I2s against the state, thereby allowing the implementation to avoid the computational cost of the CMAC function.

The drawback of this latter approach is the requirement of creating state. Finally, it also allows an implementation to use other combinations of the space-saving and computation-saving mechanisms.

Generally speaking, the puzzle mechanism works in DEX the same as in BEX. There are some implementation differences, using CMAC rather than a hash.

See Appendix A for one possible implementation. Implementations SHOULD include sufficient randomness to the algorithm so that algorithmic complexity attacks become impossible [CRO03].

4.1.2. Puzzle Exchange

The Responder starts the puzzle exchange when it receives an I1. The Responder supplies a random number I, and requires the Initiator to find a number J. To select a proper J, the Initiator must create the concatenation of the HITs of the parties and J, and feed this concatenation using I as the key into the CMAC algorithm. The lowest order K bits of the result MUST be zeros. The value K sets the difficulty of the puzzle.

To generate a proper number J, the Initiator will have to generate a number of Js until one produces the CMAC target of zeros. The Initiator SHOULD give up after exceeding the puzzle lifetime in the PUZZLE parameter ([RFC5201-bis]). The Responder needs to re-create the concatenation of the HITs and the provided J, and compute the CMAC using I once to prove that the Initiator did its assigned task.

To prevent precomputation attacks, the Responder MUST select the number I in such a way that the Initiator cannot guess it. Furthermore, the construction MUST allow the Responder to verify that the value was indeed selected by it and not by the Initiator. See Appendix A for an example on how to implement this.

Using the Opaque data field in an ECHO_REQUEST_UNSIGNED parameter ([RFC5201-bis]), the Responder can include some data in R1 that the Initiator must copy unmodified in the corresponding I2 packet. The Responder can generate the Opaque data in various ways; e.g., using some secret, the sent I, and possibly other related data. Using the same secret, the received I (from the I2), and the other related data (if any), the Receiver can verify that it has itself sent the I to the Initiator. The Responder MUST periodically change such a used secret.

It is RECOMMENDED that the Responder generates a new puzzle and a new R1 once every few minutes. Furthermore, it is RECOMMENDED that the Responder remembers an old puzzle at least 2*Lifetime seconds after

the puzzle has been deprecated. These time values allow a slower Initiator to solve the puzzle while limiting the usability that an old, solved puzzle has to an attacker.

4.1.3. HIP State Machine

The HIP protocol itself has little state. In HIP DEX, as in BEX, there is an Initiator and a Responder. Once the security associations (SAs) are established, this distinction is lost. If the HIP state needs to be re-established, the controlling parameters are which peer still has state and which has a datagram to send to its peer.

The HIP DEX state machine has the same states as the BEX state machine. However, there is an optional aggressive transmission feature to provide better performance in sensor networks with high packet loss. The following section documents the few differences in the DEX state machine.

4.1.3.1. HIP Aggressive Transmission Mechanism

HIP DEX may be used on networks with high packet loss. DEX deals with this by using an aggressive transmission practice for I1 and I2 packets. The Initiator SHOULD continually send I1 and I2 packets at some short interval t msec, based on local policy. The transmission stops on receipt of the corresponding R1 or R2 packet, which acts as an acknowledgment receipt.

Since the Responder is stateless until it receives an I2, it does not need any special behaviour on sending R1 other than to send one whenever it receives an I1. The Responder sends an R2 after receipt every I2. The Responder does need to know that R2 was received by the Initiator. Like in BEX, the Responder can learn this when it starts receiving datagrams.

4.1.3.2. HIP States

State	Explanation
UNASSOCIATED	State machine start
I1-SENT	Initiating base exchange
I2-SENT	Waiting to complete base exchange
R2-SENT	Waiting to complete base exchange
ESTABLISHED	HIP association established
CLOSING	HIP association closing, no data can be sent
CLOSED	HIP association closed, no data can be sent
E-FAILED	HIP exchange failed

Table 1: HIP States

4.1.3.3. HIP State Processes

System behavior in state I1-SENT, Table 2.

Trigger	Action
t msec	Send I1 and stay at I1-SENT

Table 2: I1-SENT - Initiating HIP

System behavior in state I2-SENT, Table 3.

Trigger	Action
t msec	Send I2 and stay at I2-SENT

Table 3: I2-SENT - Waiting to finish HIP

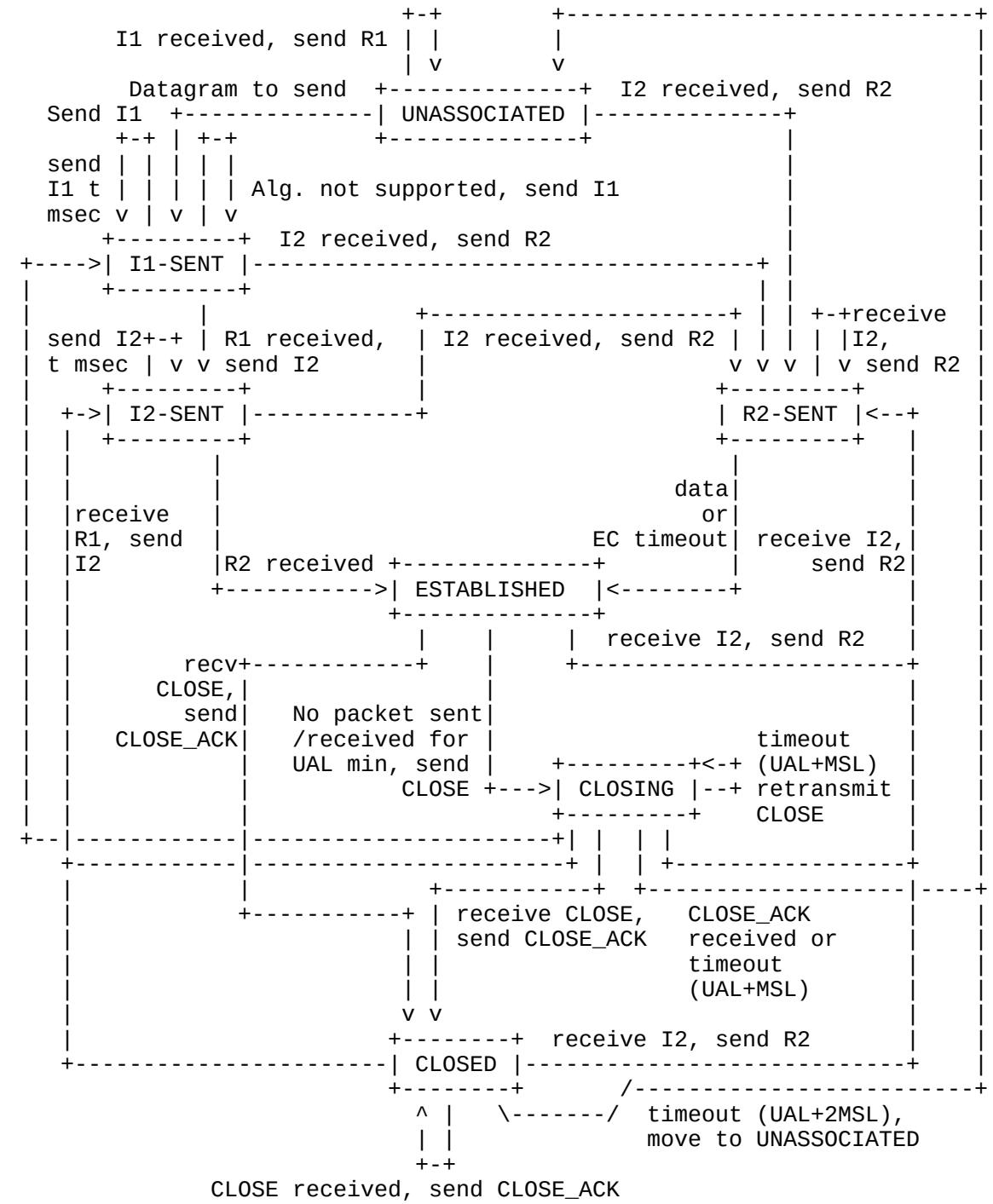
System behavior in state R2-SENT, Table 4.

Trigger	Action
Receive duplicate I2	Send R2 and stay at R2-SENT

Table 4: R2-SENT - Waiting to finish HIP

4.1.3.4. Simplified HIP State Diagram

The following diagram shows the major state transitions. Transitions based on received packets implicitly assume that the packets are successfully authenticated or processed.


I1 received, send R1
Datagram to send
Send I1
send I1 t msec
UNASSOCIATED
I2 received, send R2
Alg. not supported, send I1
I1-SENT
send I2 t msec
R1 received, send I2
I2-SENT
receive R1, send I2
R2 received
ESTABLISHED
receive I2, send R2
recv CLOSE, send CLOSE_ACK
No packet sent /received for UAL min, send CLOSE
CLOSING
timeout (UAL+MSL) retransmit CLOSE
receive CLOSE, send CLOSE_ACK
CLOSE_ACK received or timeout (UAL+MSL)
CLOSED
receive I2, send R2
timeout (UAL+2MSL), move to UNASSOCIATED
CLOSE received, send CLOSE_ACK

4.1.4. HIP DEX Security Associations

HIP DEX establishes two Security Associations (SA), one for the Diffie-Hellman derived key, or Master Key, and one for session or Pair-wise Key.

4.1.4.1. Master Key SA

The Master Key SA is used to secure DEX parameters and authenticate HIP packets. Since so little data will be protected by this SA it can be very longed lived.

The Master Key SA contains the following elements.

Source HIT

Destination HIT

HIP_Encrypt Key

HIP_MAC Key

Both keys are extracted from the Diffie-Hellman derived key via Section 6.3. Their length is determined by HIP_CIPHER.

4.1.4.2. Pair-wise Key SA

The Pair-wise Key SA is used to secure and authenticate user data. It is refreshed (or rekeyed) using the UPDATE packet exchange.

The Pair-wise Key SA elements are defined by the data transform (e.g. ESP_TRANSFORM [rfc5202-bis]).

The secrets in ENCRYPTED_KEY from I2 and R2 are concatenated to form the input to a Key Derivation Function (KDF). If the data transform does not have its own KDF, then Section 6.3 is used. Even though this input is randomly distributed, a KDF Extract phase may be needed to get the proper length for input to the KDF Expand phase.

4.1.5. User Data Considerations

There is no difference in User Data Considerations between BEX and DEX with one exception. Loss of state due to system reboot may be a critical performance issue. Thus implementors MAY choose to use non-volatile, secure storage for HIP state so that it survives system reboot. This will limit state loss during reboots to only those situtations that there is an SA timeout.

5. Packet Formats

5.1. HIP Parameters

The HIP Parameters are used to carry the public key associated with the sender's HIT, together with related security and other information. They consist of parameters, ordered according to their numeric type number and encoded in TLV format.

The following new parameter types are currently defined for DEX, in addition to those defined for BEX. Also listed are BEX parameters that have additional values for DEX.

For the BEX parameters, DIFFIE_HELLMAN, DH_GROUP_LIST, and HOST_ID, only the ECC values are valid in DEX.

TLV	Type	Length	Data
ENCRYPTED_KEY	643	variable	Encrypted container for key generation exchange
HIP_MAC_3	61507	variable	CMAC-based message authentication code
HIT_SUITE_LIST	715	variable	Ordered list of the HIT suites supported by the Responder

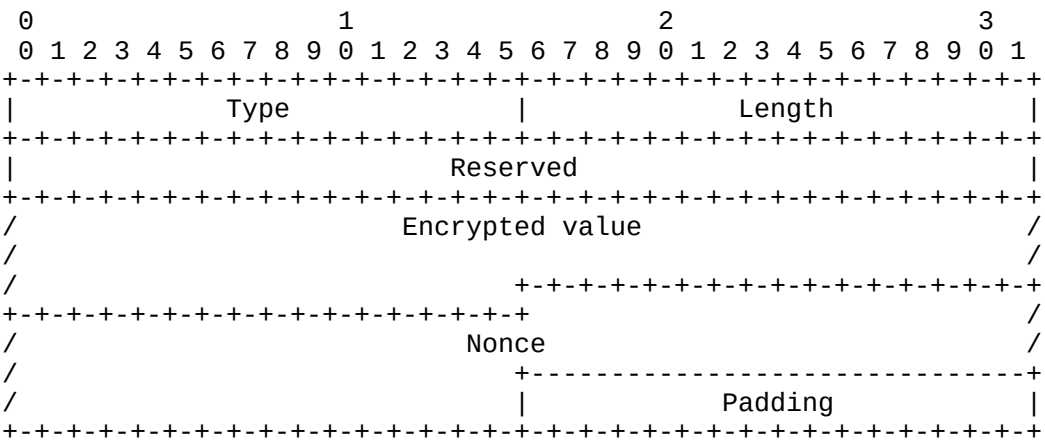
5.1.1. HIT_SUITE_LIST

The HIT suites in DEX are limited to:

HIT suite	ID
ECDH/DEX	8

The HIT_SUITE_LIST parameter contains a list of the supported HIT suite IDs of the Responder. Since the HIT of the Initiator is a DEX HIT, the Responder MUST only respond with a DEX HIT suite ID. Currently, only one such suite ID has been defined.

5.1.2. ENCRYPTED_KEY



Type	643
Length	length in octets, excluding Type, Length, and Padding
Encrypted value	The value is encrypted using an encryption algorithm as defined in the HIP_CIPHER parameter.
Nonce	Nonce included in encrypted text.

The ENCRYPTED parameter encapsulates a value and a nonce. The value is typically a random number used in a key creation process and the nonce is known to the receiver to validate successful decryption.

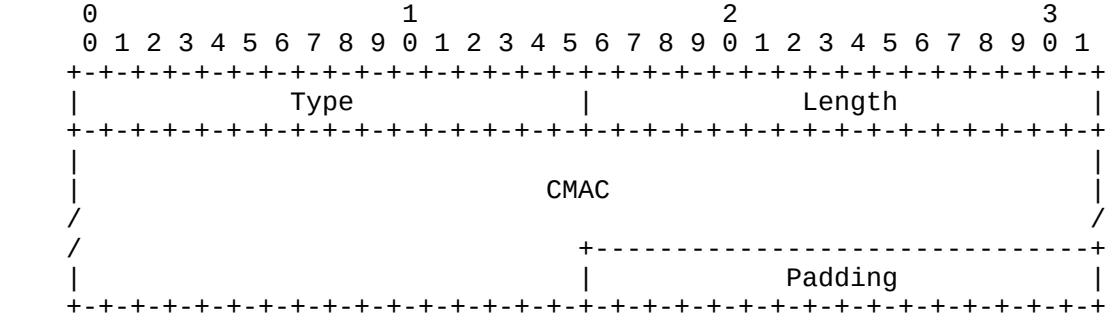
Some encryption algorithms require an IV (initialization vector). The IV MUST be known to the receiver through some source other than within the Encrypted_key block. For example the Puzzle value, I, can be used as an IV.

Some encryption algorithms require that the data to be encrypted must be a multiple of the cipher algorithm block size. In this case, the above block of data MUST include additional padding, as specified by the encryption algorithm. The size of the extra padding is selected so that the length of the unencrypted data block is a multiple of the cipher block size. The encryption algorithm may specify padding bytes other than zero; for example, AES [FIPS.197.2001] uses the PKCS5 padding scheme (see section 6.1.1 of [RFC2898]) where the remaining n bytes to fill the block each have the value n. This yields an "unencrypted data" block that is transformed to an "encrypted data" block by the cipher suite. This extra padding added to the set of parameters to satisfy the cipher block alignment rules is not counted in HIP TLV length fields, and this extra padding should be removed by the cipher suite upon decryption.

Note that the length of the cipher suite output may be smaller or larger than the length of the value and nonce to be encrypted, since the encryption process may compress the data or add additional padding to the data.

Once this encryption process is completed, the Encrypted_key data field is ready for inclusion in the Parameter. If necessary, additional Padding for 8-byte alignment is then added according to the rules of TLV Format in [RFC5201-bis].

5.1.3. HIP_MAC_3



Type	61507
Length	length in octets, excluding Type, Length, and Padding
CMAC	CMAC computed over the HIP packet, excluding the HIP_MAC parameter itself. 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 parameters when the CMAC is calculated. The size of the CMAC is the natural size of the AES block depending on the AES key size.

The CMAC calculation and verification process is presented in Section 6.2.1.

5.2. HIP Packets

DEX uses the same eight basic HIP packets (see [RFC5201-bis]) as in BEX. Four are for the HIP exchange, one is for updating, one is for sending notifications, and two are for closing a HIP association. There are some differences in the HIP parameters in the exchange packets between BEX and DEX. This section will cover the DEX packets.

An important difference between BEX and DEX HIP packets is that there

is no HIP_SIGNATURE parameter available in DEX. Thus R1 is completely unprotected and can be spoofed. The I2, R2, UPDATE, NOTIFY, CLOSE, and CLOSE_ACK parameters only have a HIP_MAC_3 parameter for packet authentication. The processing of these packets are changed accordingly.

In the future, an OPTIONAL upper-layer payload MAY follow the HIP header. The Next Header field in the header indicates if there is additional data following the HIP header. The HIP packet, however, MUST NOT be fragmented. This limits the size of the possible additional data in the packet.

5.2.1. I1 - the HIP Initiator Packet

The HIP header values for the I1 packet:

```
Header:
  Packet Type = 1
  SRC HIT = Initiator's HIT
  DST HIT = Responder's HIT, or NULL

IP ( HIP ( DH_GROUP_LIST ) )

Minimum size = 40 bytes
```

The I1 packet contains the fixed HIP header and the Initiator's DH_GROUP_LIST.

Valid control bits: none

The Initiator HIT MUST be a DEX HIT. The HIT Suite ID MUST be of a DEX type. Currently only ECDH/DEX is defined.

The Initiator receives the Responder's HIT either from a DNS lookup of the Responder's FQDN, from some other repository, or from a local table. The Responder's HIT MUST be a DEX HIT. If the Initiator does not know the Responder's HIT, it may attempt to use opportunistic mode by using NULL (all zeros) as the Responder's HIT. See also "HIP Opportunistic Mode" [RFC5201-bis].

Since this packet is so easy to spoof even if it were signed, no attempt is made to add to its generation or processing cost.

The Initiator includes a DH_GROUP_LIST parameter in the I1 to inform the Responder of its preferred DH Group IDs. Only ECDH Groups may be included in this list. Note that the DH_GROUP_LIST in the I1 packet is not protected by a MAC.

Implementations MUST be able to handle a storm of received I1 packets, discarding those with common content that arrive within a small time delta, but distinguishing this from arriving at a set time delta. This behaviour is the expected behaviour for an Initiator on a network with high packet loss. The HIP state machine calls out this behaviour in this case and the Initiator will stop sending I1 packets after it receives an R1 packet.

5.2.2. R1 - the HIP Responder Packet

The HIP header values for the R1 packet:

```
  Header:
    Packet Type = 2
    SRC HIT = Responder's HIT
    DST HIT = Initiator's HIT

  IP ( HIP ( [ R1_COUNTER, ]
             PUZZLE,
             HIP_CIPHER,
             HOST_ID,
             HIT_SUITE_LIST,
             DH_GROUP_LIST,
             [ <, ECHO_REQUEST_UNSIGNED >i ])

  Minimum size = 120 bytes
```

Valid control bits: A

If the Responder's HI is an anonymous one, the A control MUST be set.

The Initiator's HIT MUST match the one received in I1. If the Responder has multiple HIs, the Responder's HIT used MUST match Initiator's request. If the Initiator used opportunistic mode, the Responder may select freely among its HIs. See also "HIP Opportunistic Mode" [RFC5201-bis].

The R1 generation counter is used to determine the currently valid generation of puzzles. The value is increased periodically, and it is RECOMMENDED that it is increased at least as often as solutions to old puzzles are no longer accepted.

The Puzzle contains a Random #I and the difficulty K. The difficulty K indicates the number of lower-order bits, in the puzzle CMAC result, that MUST be zeros; see Section 4.1.2.

The Initiator HIT does not provide the HOST_ID key size. The Responder selects its HOST_ID based on the Initiator's preference

expressed in the DH_GROUP_LIST parameter in the I1. The Responder sends back its own preference based on which it chose the HOST_ID as DH_GROUP_LIST. This allows the Initiator to determine whether its own DH_GROUP_LIST in the I1 was manipulated by an attacker. There is a further risk that the Responder's DH_GROUP_LIST was manipulated by an attacker, as R1 cannot be authenticated in DEX as it can in BEX. Thus it is repeated in R2 allowing for a final check at that point.

In DEX, the Diffie-Hellman HOST_ID values are static. They are NOT discarded.

The HIP_CIPHER contains the encryption algorithms supported by the Responder to protect the key exchange, in the order of preference. All implementations MUST support the AES-CBC [RFC3602].

The ECHO_REQUEST_UNSIGNED contains data that the sender wants to receive unmodified in the corresponding response packet in the ECHO_RESPONSE_UNSIGNED parameter.

5.2.3. I2 - the Second HIP Initiator Packet

The HIP header values for the I2 packet:

```
  Header:
    Type = 3
    SRC HIT = Initiator's HIT
    DST HIT = Responder's HIT

  IP ( HIP ( [R1_COUNTER,]
             SOLUTION,
             HIP_CIPHER,
             HOST_ID,
             ENCRYPTED_KEY {DH, secret-x|I},
             [ ENCRYPTED {DH, ENCRYPTED_KEY {passwd, challenge } },]
             HIP_MAC_3,
             [<, ECHO_RESPONSE_UNSIGNED>i )] )

  Minimum size = 180 bytes
```

Valid control bits: A

The HITs used MUST match the ones used previously.

If the Initiator's HI is an anonymous one, the A control MUST be set.

The Initiator MAY include an unmodified copy of the R1_COUNTER parameter received in the corresponding R1 packet into the I2 packet.

The Solution contains the Random #I from R1 and the computed #J. The low-order K bits of the CMAC(S, | ... | J) MUST be zero.

In DEX, the Diffie-Hellman HOST_ID values are static. They are NOT discarded.

The HIP_CIPHER contains the single encryption transform selected by the Initiator, that will be used to protect the HI exchange. The chosen transform MUST correspond to one offered by the Responder in the R1. All implementations MUST support the AES-CBC transform [RFC3602].

The ECHO_RESPONSE_UNSIGNED contain the unmodified Opaque data copied from the corresponding echo request parameter.

The ENCRYPTED_KEY contains an Initiator generated random secret x that MUST be uniformly distributed that is concatenated with I from the puzzle. The secret x's length matches the keysize of the selected encryption transform. I from the puzzle is used as the IV in the encryption transform. This acts as a nonce from the Responder to prove freshness of the secret wrapping from the Initiator. I in the ENCRYPTED block enables the Responder to validate a proper decryption of the block. The key for the encryption is the HIP_Encrypt key.

If the Initiator has prior knowledge that the Responder is expecting a password authenication, the Initiator encrypts the ECHO_REQUEST_UNSIGNED with the password, then wraps the ENCRYPTED parameter in the secret x. I from the puzzle is used as the nonce here as well. There is no signal within R1 for this behaviour. Knowledge of password authencation must be externally configured.

The MAC is calculated over the whole HIP envelope, excluding any parameters after the HIP_MAC_3, as described in Section 6.2.1. The Responder MUST validate the HIP_MAC_3.

5.2.4. R2 - the Second HIP Responder Packet

The HIP header values for the R2 packet:

```
Header:
  Packet Type = 4
  SRC HIT = Responder's HIT
  DST HIT = Initiator's HIT

IP ( HIP ( DH_GROUP_LIST,
           ENCRYPTED_KEY {DH, secret-y|I},
           HIP_MAC_3)

Minimum size = 108 bytes
```

Valid control bits: none

The Responder repeats the DH_GROUP_LIST parameter in R2. This MUST be the same list as included in R1. The DH_GROUP_LIST parameter is repeated here because R2 is MACed and thus cannot be altered by an attacker. This allows the Initiator to determine whether its own DH_GROUP_LIST in the I1 was manipulated by an attacker.

The ENCRYPTED contains an Responder generated random secret y that MUST be uniformly distributed that is concatenated with I from the puzzle. The secret y's length matches the keysize of the selected encryption transform. I from the puzzle is used as the IV in the encryption transform. This acts as a nonce from the Initiator to prove freshness of the secret wrapping from the Responder. I in the ENCRYPTED block enables the Responder to validate a proper decryption of the block. The key for the encryption is the HIP_Encrypt key.

The HIP_MAC_3 is calculated over the whole HIP envelope, with Responder's HOST_ID parameter concatenated with the HIP envelope. The HOST_ID parameter is removed after the CMAC calculation. The procedure is described in Section 6.2.1.

The Initiator MUST validate the HIP_MAC_3.

5.3. ICMP Messages

When a HIP implementation detects a problem with an incoming packet, and it either cannot determine the identity of the sender of the packet or does not have any existing HIP association with the sender of the packet, it MAY respond with an ICMP packet. Any such replies MUST be rate-limited as described in [RFC2463]. In most cases, the ICMP packet will have the Parameter Problem type (12 for ICMPv4, 4 for ICMPv6), with the Pointer field pointing to the field that caused the ICMP message to be generated.

6. Packet Processing

Each host is assumed to have a single HIP protocol implementation that manages the host's HIP associations and handles requests for new ones. Each HIP association is governed by a conceptual state machine, with states defined above in Section 4.1.3. The HIP implementation can simultaneously maintain HIP associations with more than one host. Furthermore, the HIP implementation may have more than one active HIP association with another host; in this case, HIP associations are distinguished by their respective HITs. It is not possible to have more than one HIP association between any given pair of HITs. Consequently, the only way for two hosts to have more than one parallel association is to use different HITs, at least at one end.

6.1. Solving the Puzzle

This subsection describes the puzzle-solving details.

In R1, the values I and K are sent in network byte order. Similarly, in I2, the values I and J are sent in network byte order. The mac is created by concatenating, in network byte order, the following data, in the following order and using the CMAC algorithm with I as the key:

128-bit Initiator's HIT, in network byte order, as appearing in the HIP Payload in R1 and I2.

128-bit Responder's HIT, in network byte order, as appearing in the HIP Payload in R1 and I2.

n-bit random value J (where n is CMAC-len), in network byte order, as appearing in I2.

In order to be a valid response puzzle, the K low-order bits of the resulting CMAC MUST be zero.

Notes:

i) All the data in the CMAC input MUST be in network byte order.

ii) The order of the Initiator's and Responder's HITs are different in the R1 and I2 packets; see [RFC5201-bis]. Care must be taken to copy the values in the right order to the CMAC input.

The following procedure describes the processing steps involved, assuming that the Responder chooses to precompute the R1 packets:

Precomputation by the Responder:
Sets up the puzzle difficulty K.
Creates a R1 and caches it.

Responder:
Selects a suitable cached R1.
Generates a random number I.
Sends I and K in an R1.
Saves I and K for a Delta time.

Initiator:
Generates repeated attempts to solve the puzzle until a matching J is found:
Ltrunc(CMAC(I, HIT-I | HIT-R | J), K) == 0
Sends I and J in an I2.

Responder:
Verifies that the received I is a saved one.
Finds the right K based on I.
Computes V := Ltrunc(CMAC(I, HIT-I | HIT-R | J), K)
Rejects if V != 0
Accept if V == 0

6.2. HIP_MAC Calculation and Verification

The following subsections define the actions for processing the HIP_MAC_3 parameter.

6.2.1. CMAC Calculation

Both the Initiator and the Responder should take some care when verifying or calculating the HIP_MAC_3. Specifically, the Responder should preserve other parameters than the HOST_ID when sending the R2. Also, the Initiator has to preserve the HOST_ID exactly as it was received in the R1 packet.

The scope of the calculation for HIP_MAC_3 is:

CMAC: { HIP header | [Parameters] }

where Parameters include all HIP parameters of the packet that is being calculated with Type values from 1 to (HIP_MAC's Type value - 1) and exclude parameters with Type values greater or equal to HIP_MAC's Type value.

During HIP_MAC calculation, the following applies:

- o In the HIP header, the Checksum field is set to zero.
- o In the HIP header, the Header Length field value is calculated to the beginning of the HIP_MAC parameter.

Parameter order is described in [RFC5201-bis].

The HIP_MAC parameter is defined in Section 5.1.3. The CMAC calculation and verification process is as follows:

Packet sender:

1. Create the HIP packet, without the HIP_MAC or any other parameter with greater Type value than the HIP_MAC parameter has.
2. Calculate the Header Length field in the HIP header.
3. Compute the CMAC using either HIP-gl or HIP-lg integrity key retrieved from KEYMAT as defined in Section 6.3.
4. Add the HIP_MAC_3 parameter to the packet and any parameter with greater Type value than the HIP_MAC's (HIP_MAC_3's) that may follow.
5. Recalculate the Length field in the HIP header.

Packet receiver:

1. Verify the HIP header Length field.
2. Remove the HIP_MAC_3 parameter, as well as all other parameters that follow it with greater Type value, saving the contents if they will be needed later.
3. Recalculate the HIP packet length in the HIP header and clear the Checksum field (set it to all zeros).
4. Compute the CMAC using either HIP-gl or HIP-lg integrity key as defined in Section 6.3 and verify it against the received CMAC.
5. Set Checksum and Header Length field in the HIP header to original values.

6.3. HIP DEX KEYMAT Generation

The HIP DEX KEYMAT process is used for both the Diffie-Hellman Derived Master key and the Encrypted secrets Pair-wise key. The former uses both the Extract and Expand phases, while the later MAY

need the Extract and Expand phases if the key is longer than 128 bits. Othewise it only needs the Expand phase.

The Diffie-Hellman Derived Master key is exchanged in R1 and I2 and used in I2, R2. UPDATE, NOTIFY, and ACK packets. The Encrypted secrets Pair-wise key is not used in HIP, but is available as the datagram protection key. Some datagram protection mechanisms have their own Key Derivation Function, and if so that SHOULD be used rather than the HIP DEX KEYMAT.

The KEYMAT has two components, CKDF-Extract and CKDF-Expand. The Extract function COMPRESSES a non-uniformly distributed key, as is the output of a Diffie-Hellman key derivation, to EXTRACT all the key entropy into a fixed length output. The Expand function takes either the output of the Extract function or directly uses a uniformly distributed key and EXPANDS the length of the key, repeatedly distributing the key entropy, to produce the keys needed.

The CKDF-Extract function is following operation; the | operation denotes concatenation.

```
CKDF-Extract(DHK, info, L) -> CK
```

where

```
info    =  sort(HIT-I | HIT-R) | "CKDF-Extract"
BigK    =  Diffie-Hellman Derived or Session (x | y) Key
I       =  I from PUZZLE Parameter
```

The output CK is calculated as follows:

```
CK      = CMAC(I, BigK | info)
```

The CKDF-Expand function is following operation; the | operation denotes concatenation.

```
   CKDF-Expand(CK, info, L) -> OKM
```

where

```
     info    =  sort(HIT-I | HIT-R) | "CKDF-Expand"
     CK      =  CK from CKDF-Extract or (x | y)
     PRKlen  =  Length of PRK in octets
     maclen  =  Length of CMAC in octets = 128/8 = 16
     L        length of output keying material in octets
              (<= 255*macLen)

     If PRKlen != macLen then PRK = CMAC(0^128, PRK)
```

The output OKM is calculated as follows:

```
     N = ceil(L/macLen)
     T = T(1) | T(2) | T(3) | ... | T(N)
     OKM = first L octets of T
```

where:

```
     T(0) = empty string (zero length)
     T(1) = CMAC(CK, T(0) | info | 0x01)
     T(2) = CMAC(CK, T(1) | info | 0x02)
     T(3) = CMAC(CK, T(2) | info | 0x03)
     ...
```

(where the constant concatenated to the end of each T(n) is a single octet.)

Sort(HIT-I | HIT-R) is defined as the network byte order concatenation of the two HITs, with the smaller HIT preceding the larger HIT, resulting from the numeric comparison of the two HITs interpreted as positive (unsigned) 128-bit integers in network byte order.

x and y values are from the ENCRYPTED parameters from I2 and R2 respectively.

The initial keys are drawn sequentially in the order that is determined by the numeric comparison of the two HITs, with comparison method described in the previous paragraph. HOST_g denotes the host with the greater HIT value, and HOST_l the host with the lower HIT value.

The drawing order for initial keys:

HIP-gl encryption key for HOST_g's outgoing HIP packets

HIP-gl integrity (CMAC) key for HOST_g's outgoing HIP packets

HIP-lg encryption key for HOST_l's outgoing HIP packets

HIP-lg integrity (CMAC) key for HOST_l's outgoing HIP packets

The number of bits drawn for a given algorithm is the "natural" size of the keys. For the mandatory algorithms, the following sizes apply:

AES 128 or 256 bits

If other key sizes are used, they must be treated as different encryption algorithms and defined separately.

6.4. Processing Incoming I1 Packets

An implementation SHOULD reply to an I1 with an R1 packet, unless the implementation is unable or unwilling to set up a HIP association. An I1 in DEX is handled identically to BEX with the exception that in constructing the R1, the Responder SHOULD select a HIT that is constructed with the MUST algorithm, which is currently ECDH.

6.4.1. R1 Management

All compliant implementations MUST produce R1 packets. An R1 in DEX is handled identically to BEX.

6.5. Processing Incoming R1 Packets

A system receiving an R1 MUST first check to see if it has sent an I1 to the originator of the R1 (i.e., it is in state I1-SENT). An R1 in DEX is handled identically to BEX with the following differences.

If the system has been sending out a stream of I1 packets to work around high packet loss on a network, it stops sending the I1 packets AFTER successfully processing a R1 packet.

There is no HIP_SIGNATURE in the R1 packet. It is an unauthentication packet.

The following steps define the conceptual processing rules for responding to an R1 packet that are different than in BEX:

1. If the system is configured with an authentication password for the responder, it constructs the authentication response to

include in the I2.

2. The system prepares and sends an I2, as described in Section 5.2.3. The system MAY be configured to continually send this I2 until it receives and validates an R2.

6.6. Processing Incoming I2 Packets

Upon receipt of an I2, the system MAY perform initial checks to determine whether the I2 corresponds to a recent R1 that has been sent out, if the Responder keeps such state. An I2 in DEX is handled identically to BEX with the following differences.

The HIP implementation SHOULD process the I2. This includes validation of the puzzle solution, extracting the ENCRYPTED key for processing I2, decrypting the Initiator's Host Identity, verifying the mac, creating state, and finally sending an R2.

There is no HIP_SIGNATURE on this packet. Authentication is completely based on the HIP_MAC_3 parameter.

The following steps define the conceptual processing rules for responding to an I2 packet:

1. If the system's state machine is in the I2-SENT state, the system makes a comparison between its local and sender's HITs (similarly as in Section 6.3). If the local HIT is smaller than the sender's HIT, it should drop the I2 packet, and continue using the R1 received and I2 sent to the peer earlier. Otherwise, the system should process the received I2 packet and drop any previously derived Diffie-Hellman keying material Kij and ENCRYPTED keying material it might have formed upon sending the I2 previously. The peer Diffie-Hellman key, ENCRYPTED keying material and the nonce J are taken from the just arrived I2 packet. The local Diffie-Hellman key and the nonce I are the ones that were earlier sent in the R1 packet.

2. The system MUST validate the solution to the puzzle by computing the mac described in Section 5.2.3 using the CMAC algorithm.

3. The system must extract the keying material from the ENCRYPTED parameter. This key is used to derive the HIP data keys.

4. If the checks above are valid, then the system proceeds with further I2 processing; otherwise, it discards the I2 and its state machine remains in the same state. If the system has been sending a stream of R1 packets to the HIT in the I2 the system stops sending the R1s.

6.7. Processing Incoming R2 Packets

An R2 received in states UNASSOCIATED, I1-SENT, or ESTABLISHED results in the R2 being dropped and the state machine staying in the same state. If an R2 is received in state I2-SENT, it SHOULD be processed.

There is no HIP_SIGNATURE on this packet. Authentication is completely based on the HIP_MAC_3 parameter.

The conceptual processing rules for an incoming R2 packet in DEX are identical to BEX with the following differences.

1. The system checks the DH_GROUP_LIST as in R1 packet processing. If the list is different from R1's there may have been a DH downgrade attack against the unprotected R1 packet. If the DH_GROUP_LIST presents a better list than recieved in the R1 packet, the system may either resend I1 within the retry bounds or abandon the HIP exchange.

2. The system must extract the keying material from the ENCRYPTED parameter. This key is concatanated with that sent in the I2 packet to form the HIP data keys.

6.8. Sending UPDATE Packets

A host sends an UPDATE packet when it updates some information related to a HIP association. DEX UPDATE handling is the similar in DEX as in BEX. The key difference is the HIP_SIGNATURE is not present.

6.9. Handling State Loss

In the case of system crash and unanticipated state loss, the system SHOULD delete the corresponding HIP state, including the keying material. That is, the state SHOULD NOT be stored on stable storage. If the implementation does drop the state (as RECOMMENDED), it MUST also drop the peer's R1 generation counter value, unless a local policy explicitly defines that the value of that particular host is stored. An implementation MUST NOT store R1 generation counters by default, but storing R1 generation counter values, if done, MUST be configured by explicit HITs.

7. HIP Policies

There are a number of variables that will influence the HIP exchanges that each host must support. All HIP implementations MUST support more than one simultaneous HI, at least one of which SHOULD be

reserved for anonymous usage. Although anonymous HIs will be rarely used as Responders' HIs, they will be common for Initiators. Support for more than two HIs is RECOMMENDED.

Many Initiators would want to use a different HI for different Responders. The implementations SHOULD provide for an ACL of Initiator's HIT to Responder's HIT. This ACL SHOULD also include preferred transform and local lifetimes.

The value of K used in the HIP R1 packet can also vary by policy. K should never be greater than 20, but for trusted partners it could be as low as 0.

Responders would need a similar ACL, representing which hosts they accept HIP exchanges, and the preferred transform and local lifetimes. Wildcarding SHOULD be supported for this ACL also.

8. Security Considerations

HIP is designed to provide secure authentication of hosts. HIP also attempts to limit the exposure of the host to various denial-of-service and man-in-the-middle (MitM) attacks. In so doing, HIP itself is subject to its own DoS and MitM attacks that potentially could be more damaging to a host's ability to conduct business as usual.

HIP DEX replaces the SIGMA authenticated Diffie-Hellman key exchange of BEX with a random generated key exchange encrypted by a Diffie-Hellman derived key. Both the Initiator and Responder contribute to this key.

The strength of the key is based on the quality of the secrets generated the Initiator and Responder. Since the Initiator is commonly a sensor there is a natural concern about the quality of its random number generator.

DEX lacks Perfect Forward Secrecy (PFS). If the Initiator's HI is compromised, ALL HIP connections protected with that HI are compromised.

The puzzle mechanism using CMAC may need further study that it does present the desired level of difficulty.

The DEX HIT extraction MAY present new attack opportunities; further study is needed.

The R1 packet is unprotected and offers an attacker new resource attacks against the Initiator. This is mitigated by the Initator

only processing a received R1 when it has sent an I1. This is another DoS attack, but for battery powered Initiators, it could be a concern.

9. IANA Considerations

IANA has reserved protocol number 139 for the Host Identity Protocol.

The following HIT suites are defined for DEX HIT generation.

Index	Hash function	Signature algorithm family	Description
5	LTRUNC	ECDH	ECDH HI truncated to 96 bits

Table 5: HIT Suites

10. Acknowledgments

The drive to put HIP on a cryptographic 'Diet' came out of a number of discussions with sensor vendors at IEEE 802.15 meetings. David McGrew was very

11. References

11.1. Normative References

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

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC2463] Conta, A. and S. Deering, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 2463, December 1998.

[RFC3602] Frankel, S., Glenn, R., and S. Kelly, "The AES-CBC Cipher Algorithm and Its Use with IPsec", RFC 3602, September 2003.

[RFC3972] Aura, T., "Cryptographically Generated

Addresses (CGA)", RFC 3972, March 2005.

[RFC4309] Housley, R., "Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)", RFC 4309, December 2005.

[RFC4843-bis] Laganier, J. and F. Dupont, "An IPv6 Prefix for Overlay Routable Cryptographic Hash Identifiers (ORCHID)", draft-ietf-hip-rfc4843-bis-00 (work in progress), August 2010.

[RFC5201-bis] Moskowitz, R., Heer, T., Jokela, P., and T. Henderson, "Host Identity Protocol Version 2 (HIPv2)", draft-ietf-hip-rfc5201-bis-05 (work in progress), March 2011.

[RFC6090] McGrew, D., Igoe, K., and M. Salter, "Fundamental Elliptic Curve Cryptography Algorithms", RFC 6090, February 2011.

[rfc5202-bis] Jokela, P., Moskowitz, R., Nikander, P., and J. Melen, "Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP)", draft-ietf-hip-rfc5202-bis-00 (work in progress), September 2010.

11.2. Informative References

[AUR03] Aura, T., Nagarajan, A., and A. Gurtov, "Analysis of the HIP Base Exchange Protocol", in Proceedings of 10th Australasian Conference on Information Security and Privacy, July 2003.

[CRO03] Crosby, SA. and DS. Wallach, "Denial of Service via Algorithmic Complexity Attacks", in Proceedings of Usenix Security Symposium 2003, Washington, DC., August 2003.

[FIPS.197.2001] National Institute of Standards and Technology, "Advanced Encryption Standard (AES)", FIPS PUB 197, November 2001, <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>.

[IEEE.802-11.2007] "Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific

requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", IEEE Standard 802.11, June 2007, <http://standards.ieee.org/getieee802/download/802.11-2007.pdf>.

[IEEE.802-15-4.2006] "Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)", IEEE Standard 802.15.4, September 2006, <http://standards.ieee.org/getieee802/download/802.15.4-2006.pdf>.

[RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[RFC2898] Kaliski, B., "PKCS #5: Password-Based Cryptography Specification Version 2.0", RFC 2898, September 2000.

[RFC4306] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005.

[rfc4423-bis] Moskowitz, R., "Host Identity Protocol Architecture", draft-ietf-hip-rfc4423-bis-02 (work in progress), February 2011.

Appendix A. Using Responder Puzzles

As mentioned in Section 4.1.1, the Responder may delay state creation and still reject most spoofed I2s by using a number of pre-calculated R1s and a local selection function. This appendix defines one possible implementation in detail. The purpose of this appendix is to give the implementors an idea on how to implement the mechanism. If the implementation is based on this appendix, it MAY contain some local modification that makes an attacker’s task harder.

The Responder creates a secret value S, that it regenerates periodically. The Responder needs to remember the two latest values of S. Each time the S is regenerated, the R1 generation counter value is incremented by one and the Responder generates an R1 packet.

When the Initiator sends the I1 packet for initializing a connection, the Responder gets the HIT and IP address from the packet, and

generates an I value for the puzzle.

```
I value calculation:
I = Ltrunc( CMAC ( S, HIT-I | HIT-R | IP-I | IP-R ), n)
where n = CMAC-len
```

From an incoming I2 packet, the Responder gets the required information to validate the puzzle: HITs, IP addresses, and the information of the used S value from the R1_COUNTER. Using these values, the Responder can regenerate the I, and verify it against the I received in the I2 packet. If the I values match, it can verify the solution using I, J, and difficulty K. If the I values do not match, the I2 is dropped.

```
puzzle_check:
V := Ltrunc( CMAC( I2.I | I2.I,  I2.hit_i | I2.hit_r | I2.J ), K )
if V != 0, drop the packet
```

If the puzzle solution is correct, the I and J values are stored for later use. They are used as input material when keying material is generated.

Keeping state about failed puzzle solutions depends on the implementation. Although it is possible for the Responder not to keep any state information, it still may do so to protect itself against certain attacks (see Section 4.1.1).

Appendix B. Generating a Public Key Encoding from an HI

The following pseudo-code illustrates the process to generate a public key encoding from an HI for ECDH.

Author’s Address

Robert Moskowitz
Verizon Telcom and Business
1000 Bent Creek Blvd, Suite 200
Mechanicsburg, PA
USA

EMail: robert.moskowitz@verizonbusiness.com

Internet Engineering Task Force J. Pellikka
Internet-Draft Centre for Wireless
Intended status: Experimental Communications, University of Oulu
Expires: May 16, 2011 November 12, 2010

HIP Certificate Requests
draft-pellikka-hiprg-certreq-00

Abstract

This memorandum describes how the HIP control packets can be used to send requests for preferred certificates to the correspondent hosts. This document specifies a new CERTREQ parameter and describes its use in requesting an issuance of certificates from accepted Certification Authorities (CAs). This document focuses on the means as to how to request for a certificate, and how to signal an error in case of the desired certificate is not available. The syntax of certificates and certificate requests is out of scope of this document.

Requirements Language

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 [RFC2119].

Status of this Memo

Copyright Notice

1. Introduction

Certificate requests are authorized requests for an authority to issue a certificate, which binds the identity and other relevant information on the requestor to its public key. These requests typically contain the public key of the requestor along with registration information to be associated with the certificate by a trusted Certification Authority (CA).

Host Identity Protocol (HIP) [RFC5201] is a mobility and multihoming protocol, which separates the end-point-identifier and locator roles of IP address. The protocol introduces a new cryptographic namespace based on the public/private key cryptography. As HIP-capable hosts are effectively identified by public keys and they are able to sign information with their private key, their identity along with other host related information can be verified by a trusted 3rd party.

This memorandum describes how HIP control packets can be harnessed to transmit requests for desired type of certificates of an authority accepted by the requestor. The document specifies a new CERTREQ parameter to convey the certificate requests and describes its combined use with the CERT parameter, a placeholder for the certificate and certificate request related data. How to use the CERT parameter to convey digital certificates is specified in [I-D.ietf-hip-cert].

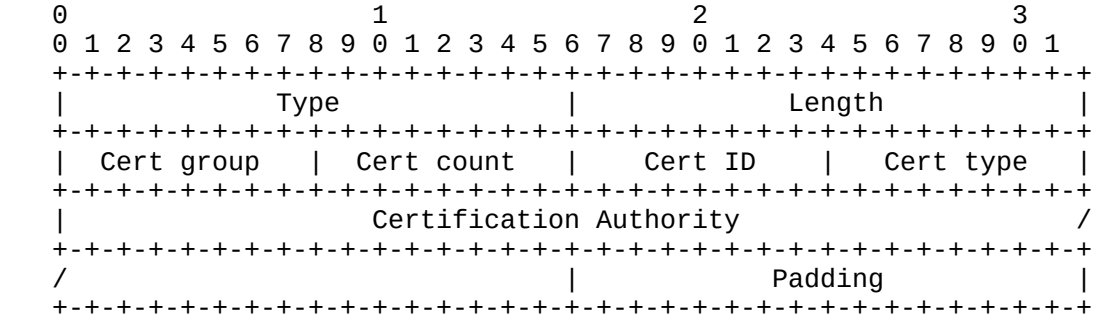
2. CERTREQ Parameter

The CERTREQ parameter provides HIP hosts with a means to request preferred certificates via the HIP control packets. The CERTREQ parameter MAY be included in the HIP Base Exchange (BEX) packets (i.e. I1, R1, I2, R2) or other HIP control packets transmitted after the communicating hosts have successfully authenticated one another. It is, however, NOT RECOMMENDED to include a CERTREQ parameter in the

I1 packet, nor it is NOT RECOMMENDED to process the parameter if present in the I1 packet.

A bit confusingly, the CERTREQ parameter is not to hold the actual certificate request, but instead carries the type of the requested certificate and a list of accepted CAs for it. The request for the desired certificate values is conveyed inside the CERT parameter. The syntax of this request is certificate type specific and thus is not described in this document. The CERTREQ parameter is included in HIP SIGNATURE, when present in the HIP packet.

The CERTREQ parameter is defined below. The fields Cert group, Cert count, Cert ID, and Cert type has been previously defined in [I-D.ietf-hip-cert] and their use conforms to what has been described in that document. The use case described by this document, however, maps the CERT and CERTREQ parameters in the same logical group by sharing the same value in the Cert group field.



Type	770
Length	Size in octets, excluding Type, Length, and Padding
Cert group	Group ID grouping multiple related CERTREQ parameters
Cert count	Total number of CERTREQ parameters in one request
Cert ID	Sequence number for this CERTREQ parameter
Cert type	Indicates the type for the requested certificate
Padding	To make the TLV a multiple of 8 bytes (if needed)

As CERT and CERTREQ parameters are related, the Cert group field provides a means to define the two parameters as belonging to the same request. The value in the Cert ID field uniquely identifies a CERTREQ parameter in the group, while the Cert count indicates the total number of CERTREQ field belonging to the certificate request. In other words, the namespace of Cert IDs are separate between the CERT and CERTREQ parameters. In order to transmit a successful request, at least one CERTREQ and one CERT parameter sharing the same value in their Cert group fields MUST be transmitted. The Cert count field is set to indicate the total count of CERTREQ parameters

belonging to the request. The values of the Cert ID and Cert group fields MUST start from 1 and their namespaces are locally managed by the host transmitting certificate requests in the HIP control packets.

The Certificate Type field has the same values as those defined in [I-D.ietf-hip-cert]. The Certificate Authority field contains an indicator of trusted CAs for the certificate type. The field contains a concatenation of all accepted CAs by order of preference, listing the most preferred CA first. The encoding of each CA indication is a SHA1 hash over the public key indicated in the certificate of the CA in question. The hashes are appended without any delimiters or other formatting. If the certificate type does not explicitly allow a concatenated list as a payload, each accepted CA MAY be included in separate CERTREQ parameters carried by the same or sequential HIP control packets. Also in this case, the CAs are listed by order of preference using the Cert ID field as an identifier.

3. Error Signaling

If the requestee is prevented from delivering the desired kind of certificate to the requestor, it MAY signal this by transmitting a HIP NOTIFY packet with the error type of the NOTIFICATION parameter set to CERTIFICATE_NOT_AVAILABLE. The CERTIFICATE_NOT_AVAILABLE error type can be carried in the NOTIFICATION parameter of other HIP packets as well. The CERTIFICATE_NOT_AVAILABLE error type is defined as follows.

NOTIFICATION PARAMETER - ERROR TYPES	Value
CERTIFICATE_NOT_AVAILABLE	52

Transmitted if the requestee is not able to deliver a certificate of the requested type or a certificate issued by any of the CAs accepted by the requestor. Notification Data contains an octet, i.e. Cert group to identify the request that the requestee was not able not fulfil.

The CERTREQ parameter should not be considered as a mandate for a certain type of certificate, but merely as a suggestion from the requestor. Therefore, if the requestee is unable to deliver the desired kind of certificate, this SHOULD NOT necessarily be considered as error and the operation of the HIP protocol SHOULD proceed as normal. In some cases where, e.g. a HIP host is

configured to require a certificate for a successful authentication and authorization, it MAY, however, be required to terminate the BEX or ongoing HIP session with the correspondent host if desired credentials cannot be obtained.

4. IANA Considerations

This memorandum specifies the CERTREQ parameter for Host Identity Protocol (HIP) [RFC5201]. The parameter is defined in Section 2 of this document and identified by type 770. The assigned type number needs to be confirmed and included to the "Host Identity Protocol (HIP) Parameters" registry by IANA.

The CERTREQ parameter contains a 8-bit unsigned integer field for a certificate type. The certificate types are maintained in a sub-registry referred to as "HIP certificate types" under the "Host Identity Protocol (HIP) Parameters" by IANA. This document conforms with the certificate type values defined by [I-D.ietf-hip-cert] and does not specify any additional values. It is assumed that the values for certificate types are maintained in that particular document.

Finally, in Section 3, this memorandum specifies a new type CERTIFICATE_NOT_AVAILABLE for the "NOTIFY message types" sub-registry under "Host Identity Protocol (HIP) Parameters". Its value is specified as 52.

5. Security Considerations

The CERTREQ parameter SHOULD NOT be attached to the I1 packet of BEX. If the Responder was to receive an I1 packet with a CERTREQ parameter, it SHOULD ignore it and proceed with the BEX as normal. The handling of CERTREQ parameter requires the Responder to utilize CPU and memory, and therefore handling the parameter before the correspondent host is authenticated would allow a Denial of Service (DoS) attack toward the Responder.

6. Acknowledgements

The authors would like to thank A. Gurtov for fruitful conversations on the subject of this document.

7. Normative References

[I-D.ietf-hip-cert]
Heer, T. and S. Varjonen, "Host Identity Protocol Certificates", draft-ietf-hip-cert-05 (work in progress), November 2010.

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

[RFC5201] Moskowitz, R., Nikander, P., Jokela, P., and T. Henderson, "Host Identity Protocol", RFC 5201, April 2008.

Author's Address

Jani Pellikka
Centre for Wireless Communications, University of Oulu
P.O. Box 4500, FI-90014
Oulu,
Finland

Phone: +358 8 553 2965
Email: jani.pellikka@ee.oulu.fi
URI: http://www.cwc.oulu.fi