Host Identity Protocol Extensions for the Traversal of Network Address Translators

Vivien Schmitt

Master Thesis

May 2006

Advisor
Prof. Dr. Jacques Calmet

Institute for Algorithms and Cognitive Systems
University of Karlsruhe (TH)

Supervisors
Dr. Marcus Brunner
Dr. Lars Eggert
Martin Stiemerling

NEC Europe Ltd.
Network Laboratories, Heidelberg

Statement of Originality

Hereby, I declare this thesis to be the work of my own, written independently. All sources and references are correctly cited with complete references to their sources.

Karlsruhe, May 23, 2006.

Acknowledgements

In the first place, I would like to thank Prof. Dr. Jacques Calmet who allowed me to achieve my master thesis at NEC Network Laboratories.

I would also like to thank my supervisors at NEC, first for welcoming me in the NGI group and offering me an interesting and creative project, and second for their help and guidance during the past six months.

Thanks also to Miika Komu and Abhinav Pathak for their collaboration.

I shall not forget to thank all the students from NEC, with whom I have spent a really nice time in Heidelberg, not only at the office but also during our free time. And more generally, thanks a lot to all of those who made me really enjoy these last few months in Germany !

Abstract

In the current Internet architecture, communications are based on IP addresses. IP addresses serve both as identifiers for applications, and topological locators for data transmissions. This dual role is however becoming problematic, in particular in mobility scenarios. The Host Idendity Protocol (HIP) introduces a new namespace, composed of Host Identities (HIs), which allows the separation of these roles. While IP addresses are still used as locators to transmit data, HIs take the role of identifiers in applications.

HIP communications are based on two different protocols. First, the HIP protocol itself, for the establishment and management of communications. Second, a security protocol, which is usually IPsec ESP, to exchange user and application data in a secured way.

To enable HIP communications over the public Internet, the protocol must be adpated to the possible presence of middleboxes, and especially Network Address Translators (NATs). NATs are likely to prevent the proper transmission of HIP and ESP traffic. For this reason, specific measures must be defined in the protocol to allow NAT traversal.

The present thesis proposes a protocol extension which provides NAT traversal support for HIP communications. This extension is based on UDP-encapsulation mechanisms, the use of Bound End-to-End Tunnel (BEET) mode ESP Security Associations, and relay mechanisms with an enhanced Rendezvous Server (RVS).

Contents

Chapter 1

Introduction

In this introduction, we will first describe the general context of the thesis. We will then introduce the new Host Identity Protocol (HIP) and the related issue that the present thesis proposes to solve.

1.1 General Network Architecture

In current networks, communications are usually described with the ISO/OSI Network model. This model is composed of seven layers presented in Figure 1.1. Each layer uses functions from lower layers and provides new functionalities to upper layers. Communication protocols are generally associated to one specific layer. In particular the third and forth layers play an important role in Internet communications. The third layer, designated as *Network Layer*, is usually managed by the Internet Protocol (IP, [1] and [2]). The forth layer, designated as *Transport Layer*, handles the data segments which are transmitted in IP packets. Typical protocols for the transport layer are Transmission Control Protocol (TCP, [3]) or User Datagram Protocol (UDP, [4]).

The Internet Protocol defines one of the two main namespaces currently used in the Internet, the namespace composed of IP addresses. The second main namespace is composed of Domain Name System (DNS) names. While DNS names are used as identifiers on application level, IP addresses have a larger role and are used in the ISO/OSI Network model from the network layer to the application layer. From the network layer point of view, an IP address is bound to a network interface and is used to determine the location of this interface in the whole network. If a host modifies its network connection, its IP address will change accordingly to represent the location of the new interface. This property can be described as the *locator* role of an IP address. IP addresses are however also used in upper layers and designate in that case the end-hosts of a communication. As such, IP addresses are not meant to change during ongoing communications. This second

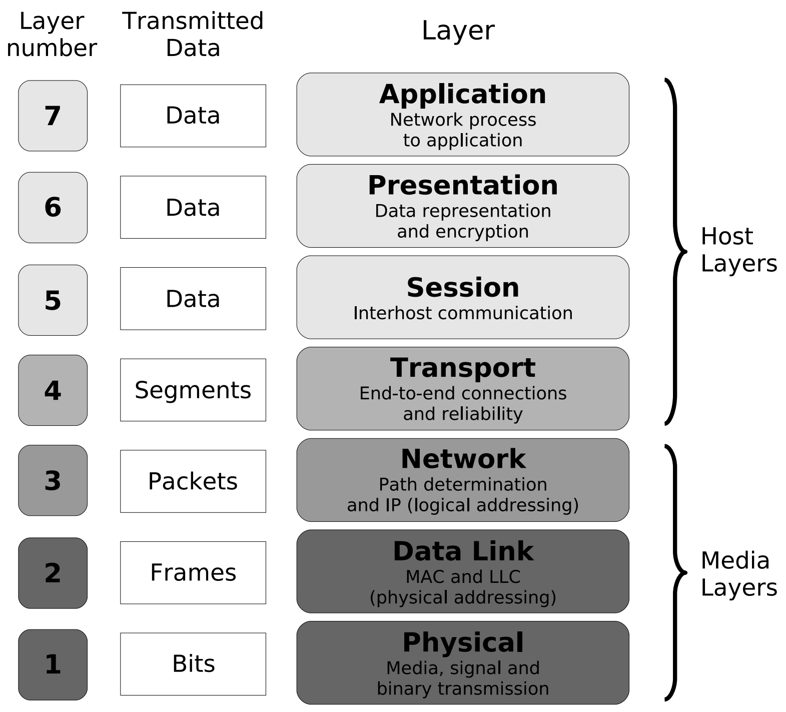
Layer number
Transmitted Data
Layer
7
Data
Application
Network process to application
6
Data
Presentation
Data representation and encryption
5
Data
Session
Interhost communication
4
Segments
Transport
End-to-end connections and reliability
3
Packets
Network
Path determination and IP (logical addressing)
2
Frames
Data Link
MAC and LLC (physical addressing)
1
Bits
Physical
Media, signal and binary transmission
Host Layers
Media Layers

Figure 1.1: ISO/OSI seven-layer network model.

property is referred to as the *identifier* role of IP addresses. Thus IP addresses have a dual role of locators and identifiers.

1.2 Introduction of a New Namespace

The dual role of IP addresses previously mentioned, is becoming problematic for several reasons. First, the version 4 of the Internet Protocol, based on 32-bit addresses, remains very widely used in comparison to version 6. Therefore, the IP address namespace suffers from the increasing number of hosts connected to the Internet and the lack of IPv4 addresses. Furthermore, the increasing mobility of devices connected to the Internet implies new requirements which are not compatible with the duality constraint of IP addresses. Multihomed services can also encounter difficulties due to this duality.

A reflexion to solve these problems has led to the development of the Host Identity Protocol (HIP). HIP introduces a new namespace composed of Host Identities (HIs). A Host Identity is a cryptographic entity which corresponds to an asymmetric key-pair. The public identifier associated to a HI is consequently the public key of the key-pair. Each host may own several HIs, but a given

HI is uniquely bound to a single host.

The new identity domain introduced by HIP enables the separation of the roles of IP addresses. While IP addresses keep their locator role in the network layer, HIs will assume the identifier role in upper layers. Therefore, considering the ISO/OSI Network model, the HIP protocol introduces a new layer between the network and transport layers (see Figure 1.2).

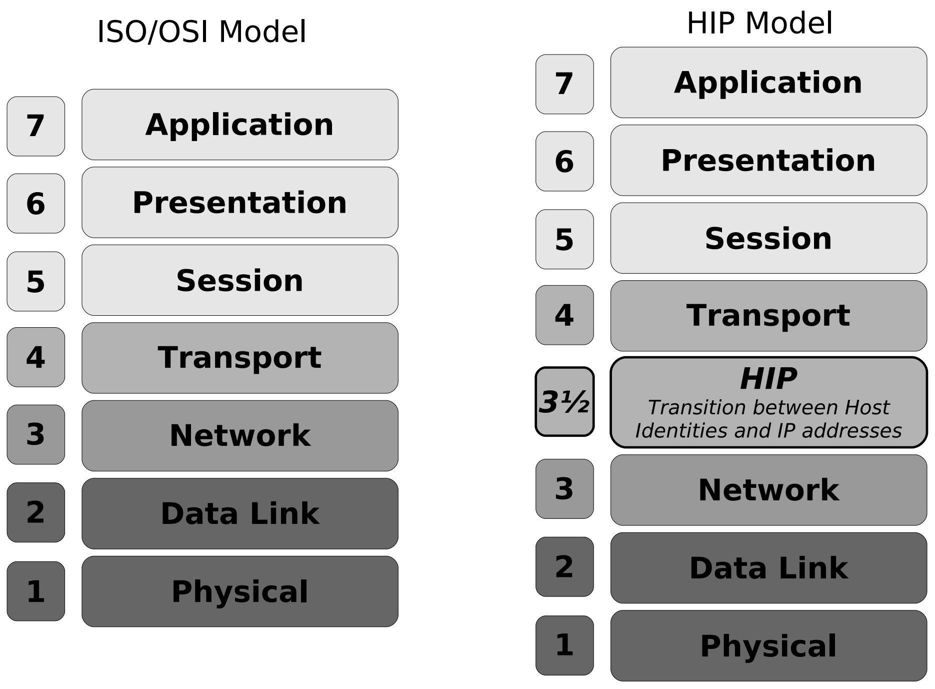

Figure 1.2: Common ISO/OSI network model and introduction of the Host Identity layer in the HIP network model.

In the HIP layer and in upper layers, Host Identifiers replace IP addresses. The conversion between a Host Identity and the corresponding IP address is established in the HIP layer. To allow legacy applications to easily use HIs instead of IP addresses, HIP defines two types of identifiers that are numerical values of the same length as common IPv4 or IPv6 addresses. The main identifiers are 128-bit Host Identity Tags (HITs) and a limited version of them are 32-bit Local Scope Identifiers (LSIs).

1.3 HIP Communications

The introduction of a new namespace and the use of the new HIP protocol implies a new way to establish communications between two hosts. HIP communications are divided in two main phases : the HIP Base Exchange and the secured data transfer.

The HIP Base Exchange uses specific HIP packets to establish a connection between two end-hosts, represented by their Host Identities. The resulting communication is therefore based on a pair of HITs or LSIs. The Base Exchange also allows the exchange and negociation of parameters and cryptographic keys for the communication.

After the Base Exchange is completed, the end-hosts can establish secured communications, based on HIs, and exchange data in a secured way. The data transfer relies on an existing end-to-end security protocol, which is typically but not necessarily the IPsec ESP protocol.

1.4 Network Structure and Middleboxes

The current Internet is organized in a multitude of interconnected networks and sub-networks. When two hosts establish a connection between each other, the traffic of their communication traverses a lot of devices which may simply route and forward it, but may also analyse and apply some modifications to it. Such devices that analyse and modify the data which go through them are designated as middleboxes. Several types of middleboxes exist, the best known being firewalls and Network Address Translators (NATs). The role of firewalls is mainly to block or forward traffic from or to a sub-network, according to a local security policy. NATs however have a more complex behavior.

One of the main purpose of NATs is to deal with the IPv4 address scarcity. Network address translation indeed consists in modifying the source and destination addresses contained in the headers of IP packets. In that way, NATs allow several hosts on a private network (and consequently configured with private IP addresses) to use a single public IP address to access the Internet. For example all the requests going from the private network to the Internet have a private address as source address in the private network. When they pass through the NAT, the source address is replaced by the public IP address of the NAT before they are forwarded to the public Internet. When replies come back to the NAT, the destination address is then replaced by the proper address of the private host which sent the request, according to a mapping table that the NAT keeps up-to-date.

There is not a single way to process network address translation. Several types of NAT exist, that follow various rules. NATs can not only modify addresses, but may also change the transport layer protocol port numbers of the data that traverse them. Another important aspect of NATs is that connections can only be initiated from the private network behind the NAT. Indeed NATs will prevent incoming connections from the public Internet, because no entry will exist beforehand in the NAT mapping table. All these features of NATs have repercussions on the end-to-end connectivity principle of the Internet.

1.5 Contribution

Since middleboxes are widely used in the current Internet, they have to be taken into account for the development of new protocols. In particular HIP communications have to take care of the possible presence of NATs between end-hosts. As previously explained, HIP communications are divided in two phases and each of them may encounter problems with NAT traversal. The first phase, the HIP base exchange, is based on specific HIP packets. Since HIP is still under development and NATs generally do not forward unknown traffic, HIP packets may be blocked by NATs. Furthermore, a well known problem of NATs consists in the invalidation of transport layer checksums which are based on IP addresses. Consequently the HIP base exchange is likely to fail due to the presence of NATs between the end-hosts. The second phase of HIP communications is based on another protocol which may also encounter difficulties with NAT traversal. This is the case in particular for the IPsec ESP protocol.

For these reasons the HIP protocol needs some enhancement to support NAT traversal. The present thesis proposes an extension of HIP providing such functionalities. In Chapter 2, a more precise description of HIP and IPsec specifications will be given. In Chapter 3, we will then present a NAT traversal solution articulated around two main points : the UDP encapsulation of transmitted data and the use of a specific IPsec ESP mode, namely the BEET mode. The extension distinguishes between two different cases due to the asymmetry of NAT behavior. First the case where only the initiator of a HIP communication is located behind a NAT, and second, the case where also the responder of the communication is protected by a NAT.

In Chapter 4, the work accomplished in a HIP implementation project, namely OpenHIP, will be presented. Eventually, some future work will be described in Chapter 5 and the thesis will be concluded in Chapter 6.

Chapter 2

Protocol Overview and Problem Description

In this chapter, we describe first the protocol specifications which are necessary to well understand the main functionalities of the corresponding protocols, the NAT traversal issue in the HIP context and the proposed extension to solve it. More details about the protocols are nonetheless available in the cited references.

After HIP and IPsec descriptions, some precisions are given about NAT functionalities and the NAT traversal issue is eventually exposed.

2.1 Host Identity Protocol

In the present section, we present the current status and main specifications of the Host Identity Protocol.

2.1.1 Current Development of HIP

The Host Identity Protocol is currently developed in two Internet Engineering Task Force (IETF) and Internet Research Task Force (IRTF) entities, namely the IETF HIP Working Group and the IRTF HIP Research Group. These groups are chartered to publish documents describing the specifications of the protocol.
The fields currently handled by these groups are following :

- HIP main architecture
- HIP Base Exchange specifications

- IPsec transport-mode ESP in HIP communications
- DNS record extension for HIP
- End-host mobility and multihoming with HIP
- HIP Rendezvous extension
- Extension for the registration to various services

A new field will be added to this list with the present work about NAT traversal for HIP communications.

In addition, several open-source projects implementing HIP specifications are currently under development.

- The OpenHIP project, for MS Windows and Linux platforms. Mainly developed by The Boeing Company.
- The InfraHIP project (previously HIPL/HIP for Linux project), for Linux operating systems. Mainly developed by the Helsinki Institute for Information Technology (HIIT).
- The HIP for inter.net project, for BSD/Linux operating systems. Mainly developed by Ericsson NomadicLab.

HIP protocol specifications are described in several IETF documents corresponding to the fields previously mentioned. In particular the main architecture of the protocol is defined in [13]. In the next sections, the main properties of the protocol will be presented. The HIP Base Exchange will first be described in Section 2.1.2. Then the secured data transfer phase will be exposed in Section 2.1.3. The two following Sections 2.1.4 and 2.1.5, will present the HIP node discovery system and the mobility management.

2.1.2 HIP Base Exchange

The first phase of a HIP communication is the HIP Base Exchange. Specifications for this phase are described in [14]. The main role of the base exchange is to establish an IP-layer communications context between two end-hosts designated by their Host Identities. This context is called *HIP association* and is kept up-to-date by procedures defined in the HIP protocol.

Both base exchange and updating procedure are based on the exchange of HIP packets between the communicating hosts. Different types of HIP packets exist but they all have the same structure described in Section 2.1.2.1. The base exchange is more precisely described in Section 2.1.2.2 and the updating procedure is exposed in Section 2.1.5.

2.1.2.1 HIP Packet Structure

All the packets used by the HIP protocol have the same structure composed of a common HIP header presented in Figure 2.1, followed by various parameters encoded in a Type-Length-Value (TLV) format.

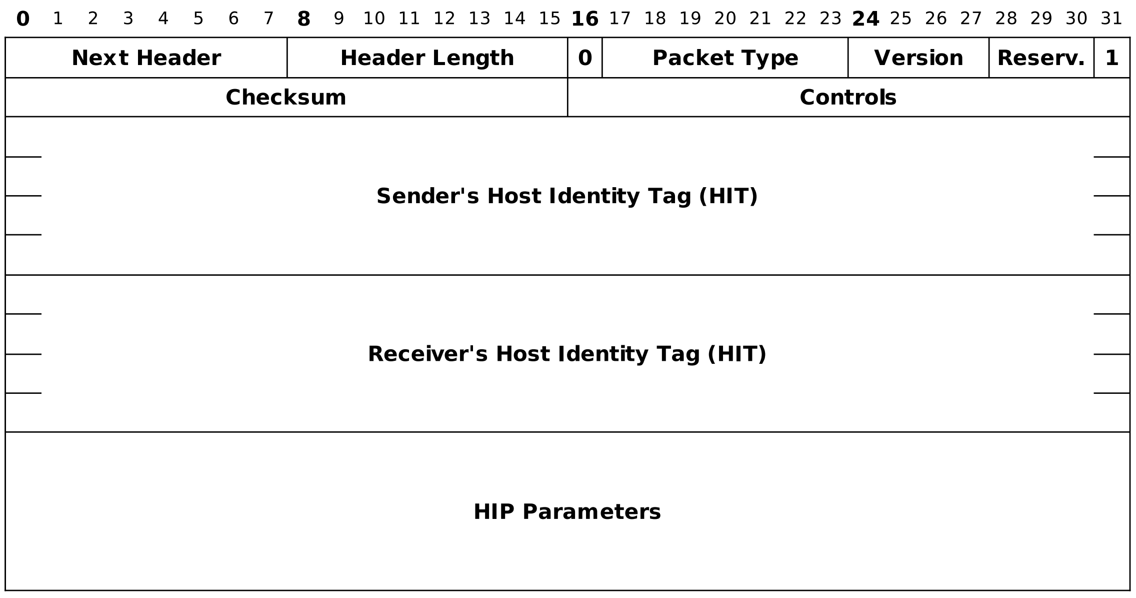

Figure 2.1: Structure of the HIP packet header.

The HIP header has the structure of an IPv6 extension header and contains the following fields :

Next Header is the common IPv6 next header protocol number.

Header Length is the total length of the HIP header and the following HIP parameters, in 8-bytes units and excluding the first eight bytes.

Packet Type is the HIP packet type number.

Version is the version number, currently 1.

Reserv. corresponds to three reserved bits, which must currently be zero.

Checksum is calculated above the whole HIP packet, including HIP parameters. It is determined in the same way as TCP or UDP checksums, using a pseudo-IPv4 or pseudo-IPv6 header containing the IP addresses of the source and destination hosts.

Controls contain information about the structure of the packet and capabilities of the host.

The HIP header also contains the two 128-bit representations of Host Identities, namely Host Identity Tags (HITs), of the packet sender and receiver. The parameters that follow the HIP header depend on the packet type.

2.1.2.2 HIP Base Exchange Procedure

The HIP Base Exchange is a four packet exchange between two end-hosts designated as *initiator* and *repsonder*. The first packet, I1, is sent by the initiator to trigger the base exchange and the creation of a HIP association. The next two packets, R1 and I2, are respectively sent by the responder and the initiator, and are used to make a Diffie-Hellman key exchange which enables the generation of a session key. These two packets also carry a puzzle and its solution. The puzzle is a cryptograhpic challenge sent by the responder, that the initiator must solve before continuing the base exchange. Finally, after validation of the I2 packet, the responder concludes the base exchange by sending the fourth packet, R2.

The last three packets of the base exchange contain a signature which enables the receiver to check the authenticity of the packet, using the sender's Host Identifier as authentication key.

The HIP base exchange is illustrated in Figure 2.2.

After the reception and validation of the R2 packet, a HIP association is established between the initiator and the responder. This association is characterized by the pair of Host Identities of the two end-hosts.

2.1.3 Secured Data Exchange over IPsec

After the HIP Base Exchange is completed, the associated hosts can in a second phase establish secured communications and exchange various data. HIP uses on this purpose an end-to-end security protocol. Typically, the user data in HIP communications are encrypted with IPsec ESP transport mode. The IPsec ESP protocol and its specificities are presented in details in Section 2.2.

In the case of IPsec ESP transport mode, authentication and encryption transforms are negociated during the HIP Base Exchange. The keys used to set up the ESP Security Associations (SAs, see Section 2.2) are drawn from the keying material generated during the Diffie-Hellman exchange

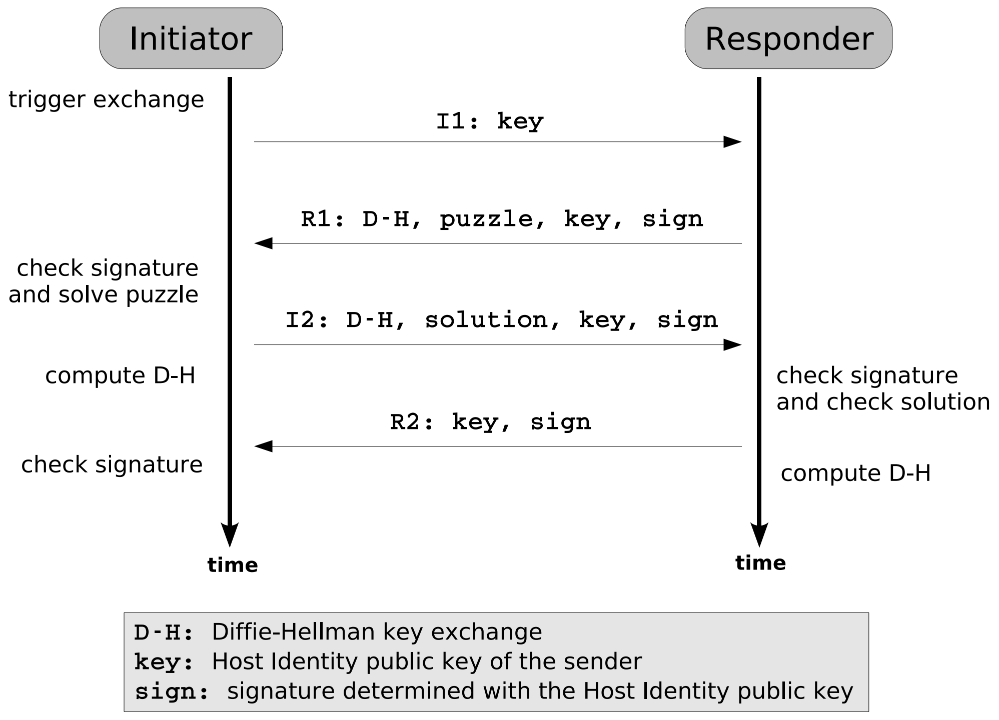
Initiator
Responder
trigger exchange
I1: key
R1: D-H, puzzle, key, sign
check signature
and solve puzzle
I2: D-H, solution, key, sign
compute D-H
check signature
and check solution
R2: key, sign
check signature
compute D-H
time
time
D-H: Diffie-Hellman key exchange
key: Host Identity public key of the sender
sign: signature determined with the Host Identity public key

Figure 2.2: HIP Base Exchange procedure.

of the base exchange. The Security Parameters Indices (SPIs, see Section 2.2) of inbound and outbound SAs are also transmitted during the base exchange. Each end-host defines the SPI of its own inbound SA and sends the value to its peer. The SPIs are transmitted in the HIP parameters of I2 and R2 packets. Then the SPIs contained in each ESP-encrypted user data packet are used to determine the HIP association to which the packet is related, because Host Identifiers are not included in these data packets. Thus the HIP base exchange plays also the role of ESP setup exchange.

IPsec ESP specifications for HIP communications are described in [15].

2.1.4 HIP Node Discovery

Before establishing a HIP communication, a host willing to communicate with another host, must first determine the Host Identity and the current IP address of this last host. Usually the initiator of the communication does not know this information beforehand. This is also the case for general communications over the Internet. Usually, the initiator knows only its peer by a Fully Qualified Domain Name (FQDN). To determine the corresponding IP address, the initiator performs first a DNS lookup. DNS servers indeed register the mapping between FQDNs and IP addresses. A DNS lookup consists therefore in sending a request containing a FQDN to a DNS server. The DNS server then sends a reply containing the IP address(es) corresponding to the requested FQDN.

To enable HIP node discovery with a similar mechanism, some improvements of DNS servers and DNS mechanisms are required. An extension of DNS records for HIP has been developed in [16]. New DNS Resource Records (RRs) are defined in this extension to enable mappings between FQDNs and HIs/HITs. Thus a HIP node, which wants to be reachable by other nodes, must register its FQDN and current IP address, and also its HI/HIT in DNS servers.

The HIP node discovery mechanism is consequently based on two DNS lookup procedures. If the initiator of a HIP communication knows only the FQDN of the host it wants to contact, it must first perform a DNS lookup to determine the HIT of its peer. The initiator sends a request, containing a FQDN and asking for the corresponding HIP record, to a DNS server. The DNS server then sends the reply containing the HI and HIT bound to the FQDN, back to the requester.

In a second step, the initiator performs a standard DNS lookup to determine the current IP address of its peer. The initiator sends a request for the IP address corresponding to the peer FQDN, to the DNS server. The server then replies with the related IP address.

Eventually, the initiator can deduce from the two DNS answers the mapping between the HIT and the IP address of its peer. Knowing both HIT and IP address of the peer, the initiator can trigger the base exchange to establish a HIP communication with it.

The double lookup mechanism is represented in Figure 2.3.

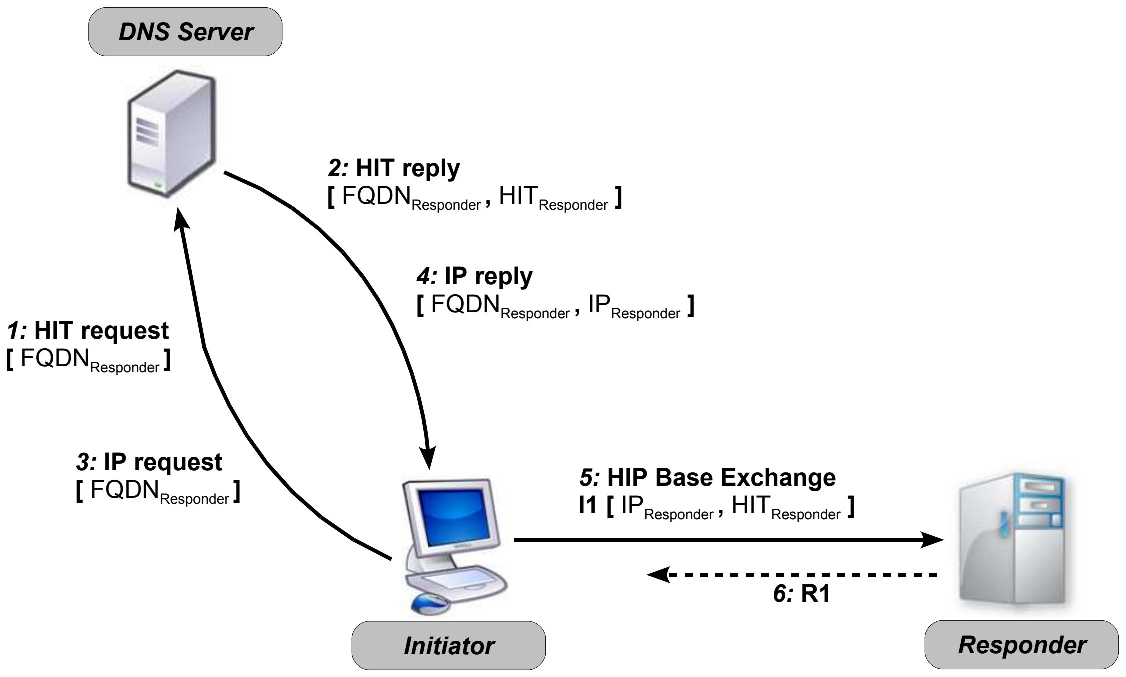

Figure 2.3: HIP node discovery. The initiator performs two DNS lookups, one to determine the HIT of the responder, the second to determine its IP address. The initiator then begins the HIP Base Exchange.

2.1.5 Mobility with HIP

One of the main purpose of HIP is to allow hosts to move and modify their network connection without resetting their ongoing communications. Since HIP decouples the identifier and locator roles of IP addresses, it can easily introduce a new mobility management. Transport layer communications are based on Host Identities, so that IP addresses can be modified without transport and upper layers being affected by this modification. The HIP protocol needs however to define a structure to update the mapping between HIs and IP addresses in the HIP layer. This is the role of HIP UPDATE packets. Specific considerations about mobility with HIP are discussed in [17, Mobility with HIP].

When a host modifies its network connection and thus its IP address, it must send a HIP UPDATE packet to its peers to alert them about its new location. This HIP UPDATE packet containing LOCATOR parameters (basically the new IP address at which the mobile host can be reached) is then acknowledged by the peers with another HIP UPDATE packet.

When IPsec ESP transport format is used to protect user data transfers, the update procedure consists in a three or four-packet exchange between the mobile host and each of its peers. This procedure is defined in [15].

Three different cases are possible, depending on the need of rekeying the ESP security associations. In every case, the mobile host must first indicate in a HIP UPDATE packet its new IP address, but also the old and new values of the incoming SA SPI. The peer then replies with another HIP UPDATE packet containing its own old and new SPI values. This second packet is sent to the new IP address of the mobile host, so that this new IP address can be checked by the peer. The IP address is validated by a third HIP UPDATE packet sent by the mobile host and which plays the role of an acknowledgement.

If no SA rekeying is processed, the update procedure is limited to this three-packet exchange and the old and new SPI values indicated in the UPDATE packets are both set up to the values of the pre-existing SA SPIs.

A second case appears if the mobile host decides to rekey the SAs during this update procedure. In such case, the new SPI values indicated in the UPDATE packets correspond to the new values chosen for the new SAs. Diffie-Hellman parameters may also be added to the UPDATE packets if new keying material needs to be generated. These two first cases are represented in Figure 2.4.

The third possible case happens when the SA rekeying procedure is decided by the peer and not the mobile host. In that case, the real new SPI values and rekeying parameters are translated to the second and third HIP UPDATE packets. Consequently a forth UPDATE packet is sent by the peer to acknowledge and conclude the update procedure. This case is represented in Figure 2.5.

The specifications previously described about the HIP protocol, are the main points to consider for the development of the NAT-traversal extension presented in the following chapters of the present thesis. The next section describes the main specifications concerning IPsec protocols.

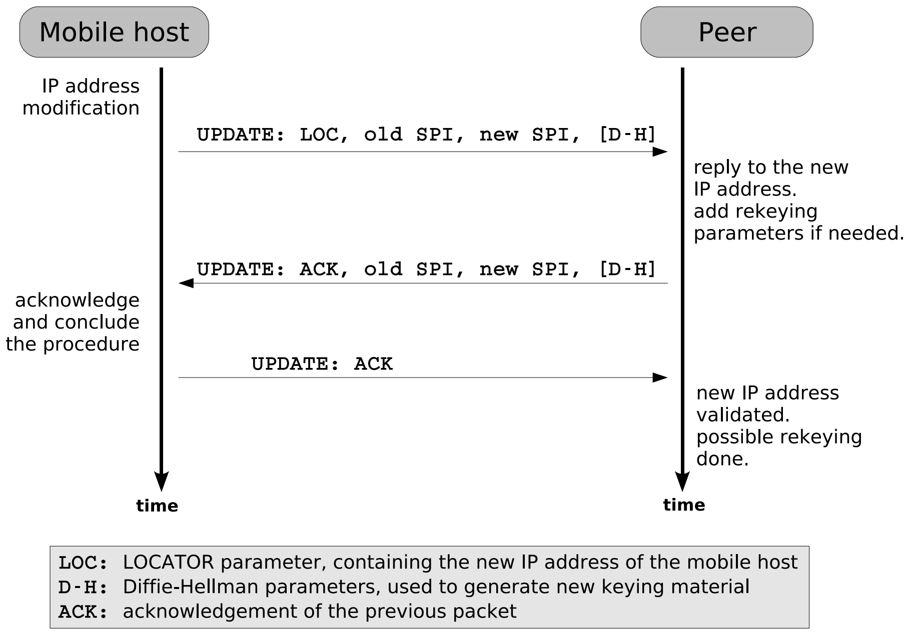
Mobile host
Peer
IP address modification
UPDATE: LOC, old SPI, new SPI, [D-H]
reply to the new IP address. add rekeying parameters if needed.
UPDATE: ACK, old SPI, new SPI, [D-H]
acknowledge and conclude the procedure
UPDATE: ACK
new IP address validated. possible rekeying done.
time
time
LOC: LOCATOR parameter, containing the new IP address of the mobile host
D-H: Diffie-Hellman parameters, used to generate new keying material
ACK: acknowledgement of the previous packet

Figure 2.4: HIP UPDATE procedure initiated by a mobile host. No SA rekeying is processed or the procedure is directly initiated by the mobile host.

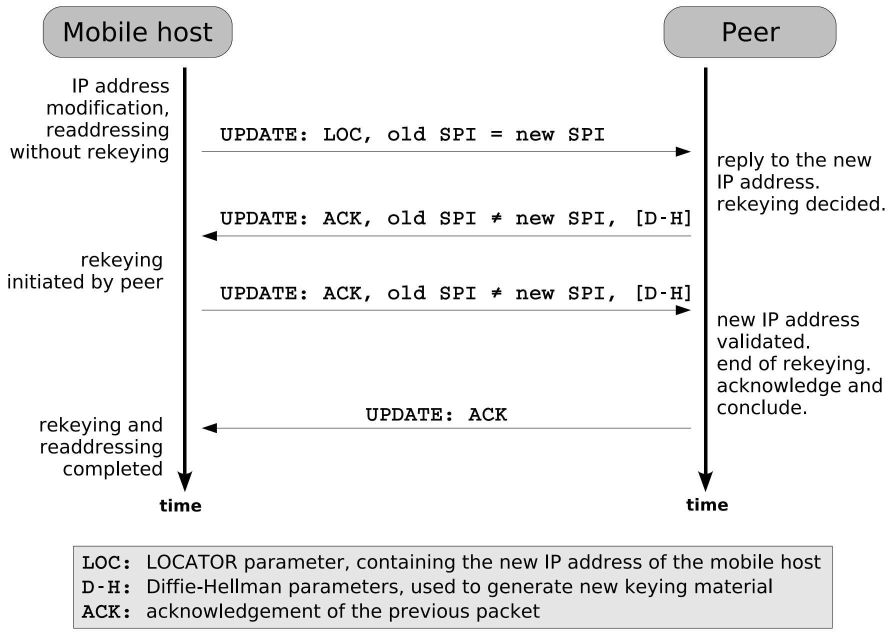
Mobile host
Peer
IP address modification, readdressing without rekeying
UPDATE: LOC, old SPI = new SPI
reply to the new IP address. rekeying decided.
UPDATE: ACK, old SPI ≠ new SPI, [D-H]
rekeying initiated by peer
UPDATE: ACK, old SPI ≠ new SPI, [D-H]
new IP address validated. end of rekeying. acknowledge and conclude.
UPDATE: ACK
rekeying and readdressing completed
time
time
LOC: LOCATOR parameter, containing the new IP address of the mobile host
D-H: Diffie-Hellman parameters, used to generate new keying material
ACK: acknowledgement of the previous packet

Figure 2.5: HIP UPDATE procedure initiated by a mobile host. SA rekeying procedure is decided by the peer.

2.2 IPsec

IPsec stands for *IP security*. It is a standard which provides security for IP communications by encrypting and/or authenticating IP packets. IPsec is based on two main cryptographic protocols, namely Encapsulating Security Payload (ESP) and Authentication Header (AH).

The main architecture of IPsec is described in [5]. ESP specifications are described in [7] and AH specifications in [6].

2.2.1 IPsec Management

IPsec protocols provide security for communications on the network layer, regardless to the transport layer protocol effectively used. To protect the transmitted data, IPsec protocols can encrypt and authenticate the IP packets with various algorithms. To determine which security protocol and which algorithm to use for a specific communication, the IPsec management is based on Security Associations (SAs).

A Security Association corresponds to an unidirectional flow of data between two specific hosts and is uniquely identified by a triple consisting of a Security Parameters Index (SPI), a destination IP address, and a security protocol (AH or ESP) identifier.

In addition, the IPsec management relies on two databases :

- the Security Association Database (SAD), which lists all the active SAs and for each SA, the encryption and/or authentication algorithms used to secure the data flow and also the necessary keys or parameters of the algorithms.
- the Security Policy Database (SPD), which contains the rules determining whether or not IPsec must be used for a specific data traffic, and if it the case, indicates the corresponding SA and security protocol to use.

As previously mentioned, AH and ESP are the main security protocol used in IPsec. While AH provides only integrity and data origin authentication for IP communications, ESP provides integrity, data origin authentication and also confidentiality (encryption) of the transmitted data. Since ESP is the main security protocol used in HIP communications, the next sections focus only on ESP specificities and do not give any detail about the AH protocol which is out of the scope of this thesis.

2.2.2 IP Encapsulating Security Payload

The ESP security protocol allows the encryption and authentication of IP packets. As already specified above, ESP can use various algorithms to encrypt and authenticate the data. Possible encryption algorithms are for example DES, 3DES, AES, Blowfish, and possible authentication algorithms are HMAC-MD5 and HMAC-SHA1. The general concept of ESP consists in encrypting the data contained in an IP packet, add an ESP header at the beginning of the packet and add an ESP trailer with an authentication field at the end of the packet. However, two different modes are defined to process ESP transform, the *transport* mode and the *tunnel* mode. These modes are respectively described in Section 2.2.2.2 and Section 2.2.2.3. In the next sub-section, we describe first the ESP packet format.

2.2.2.1 ESP Packet Format

The format of an ESP packet does not depend on the ESP mode used for a specific SA. The structure of the packet is described in Figure 2.6

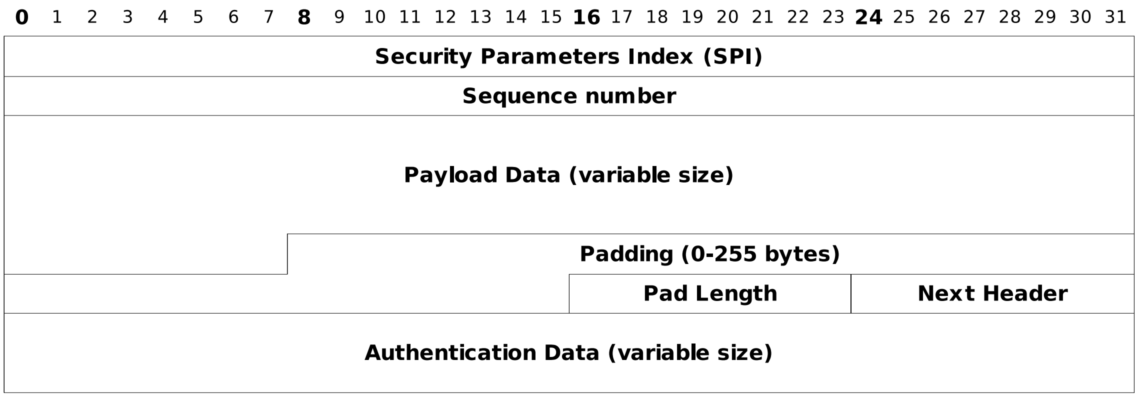

Figure 2.6: Format of a common ESP packet.

An ESP packet is composed of the following fields :

Security Parameters Index (SPI) is a 32-bit identifier which indicates in combination with the destination address of the packet and the security protocol used, to which SA the packet is related.

Sequence Number is used to verify the sequence of received ESP packets.

Payload Data contains the encrypted data which corresponds to an entire IP packet or to its payload (see following sections).

Padding is used to extend the length of the encrypted data (corresponding to the Payload Data, Padding, Pad Length and Next Header fields) to a proper length required by the encryption algorithm. The padding is used in particular when the encryption is based on a cipher block algorithm.

Pad Length indicates the number of bytes of the Padding field.

Next Header contains a protocol number which identifies the type of data contained in the Payload Data field.

Authentication Data contains an Integrity Check Value (ICV) computed over the entire ESP packet minus the present field. Its length is determined by the authentication protocol used for this ESP packet.

The SPI and Sequence Number fields constitute the ESP header of the packet, while the Padding, Pad Length and Next Header fields build the ESP trailer.

2.2.2.2 ESP Transport Mode

The ESP security protocol can be processed in two different ways. The first one is the transport mode. In this mode, the encrypted data contained in the Payload Data field of an ESP packet correspond only to the payload of the secured IP packet.

An outgoing packet is thus obtained by encrypting the payload of an IP packet and then inserting the ESP header between the original IP header and the encrypted data, and the ESP trailer at the end of the packet. Finally the authentication field is computed and put at the end of the packet.

When an incoming packet is encrypted with ESP, its decryption procedure consists in verifying the authentication data, removing the ESP header and trailer and decrypting the payload data. A new IP packet is then built with the original IP header of the incoming packet, and the decrypted data as payload.

An example of an IP packet containing TCP data, secured with ESP transport mode, is presented in Figure 2.7

2.2.2.3 ESP Tunnel Mode

A second way to apply ESP transform to an IP packet is the tunnel mode. In this mode, the ESP packet carries an entire secured IP packet. Not only the IP payload but also the IP header is

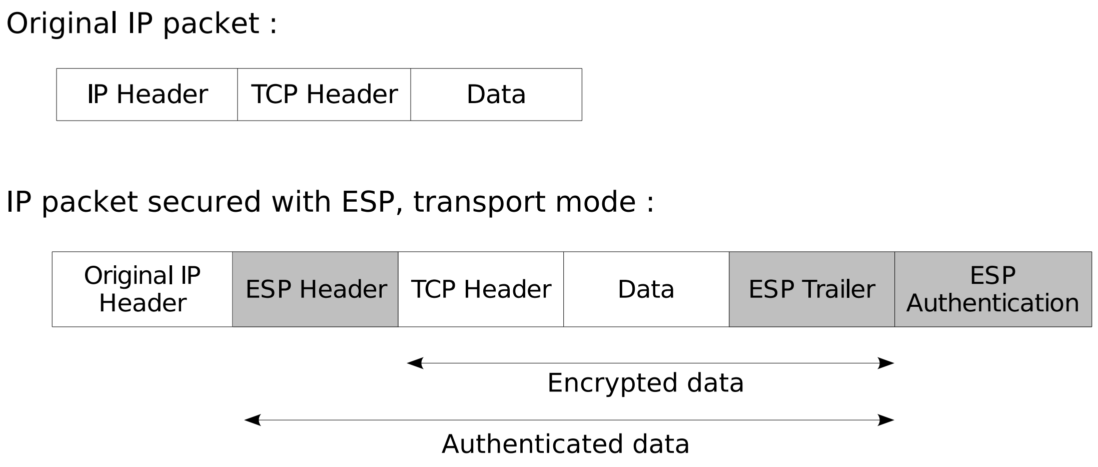

Figure 2.7: A TCP-IP packet and its corresponding secured version in ESP transport mode.

encrypted and included in the ESP payload data field. The advantage of the tunnel mode is that two different IP headers are present in ESP packets. While the external IP header can be modified by several middleboxes during the transmission of the packet, the internal IP header, included in the payload of the ESP packet, remains intact. Thus the IP addresses contained in the inner IP header are not modified during transmission. They can be used by the end-hosts as identifiers to refer to each other and furthermore the possible checksums based on IP headers and contained in the data will not be invalidated by the modification of the external IP header. The tunnel mode is consequently used when the end-hosts are not directly connected but located behind gateways or middleboxes that may modify the IP traffic between them.

The encryption procedure in tunnel mode consists in the encryption of the entire original IP packet, the creation of a new external IP header placed in front of the ESP header, the completion of the packet with the ESP trailer and finally the addition of the authentication field at the end of the packet.

The decapsulation procedure consists in the verification of the authenticity of the packet, the decryption of the payload data, and the transmission of the IP packet contained in the payload to upper network layers.

Figure 2.8 presents an IP packet with a TCP payload and the corresponding ESP packet in tunnel mode.

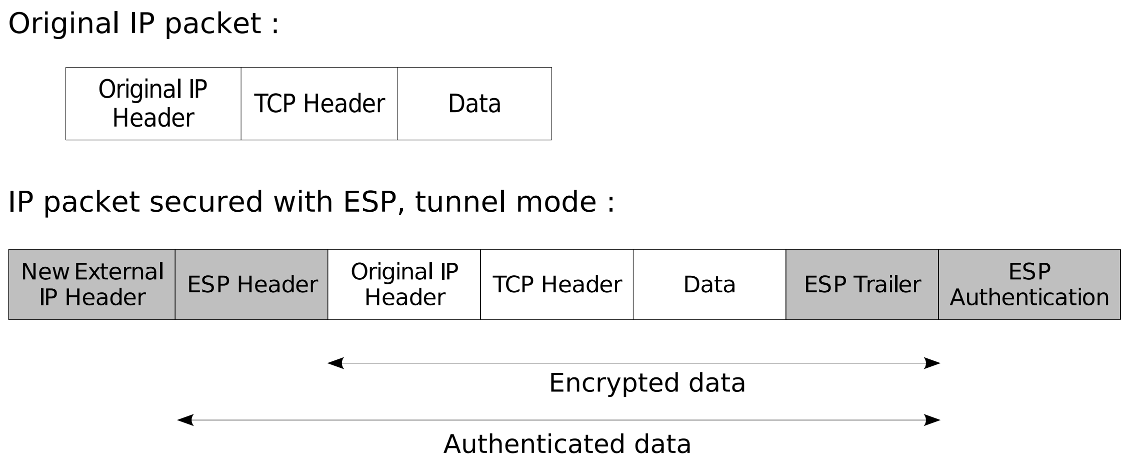

Figure 2.8: A TCP-IP packet and its corresponding secured version in ESP tunnel mode.

2.2.3 Bound End-to-End Tunnel Mode for ESP

In the previous section, the ESP protocol and its two main modes have been presented. The tunnel mode offers the possibility to avoid issues due to the modification of IP headers, by encapsulating the entire original IP packets. This however introduces an overhead in the transmitted amount of data, because each ESP packet contains two IP headers.

A new mode has been developed, that provides the same advantage as the tunnel mode but without its overhead. This new mode is called Bound End-to-End Tunnel (BEET) mode and is described in [20].

The main concept of the BEET mode is to define for each security association, two pairs of IP addresses. These pairs are respectively designated as *inner addresses* and *outer addresses*. Inner addresses are used in the upper network layers and in particular are the addresses seen and used by applications. This implies that the checksums contained in the transport layer data are computed with the inner addresses. Outer addresses are used in the network and lower layers. They are the IP addresses appearing in the wire to transmit the ESP packets. Inner addresses are strictly bound to SAs, they cannot be modified. Outer addresses can however change and be updated during the lifetime of a SA. Furthermore, inner and outer addresses do not need to be from the same IP family. Inner addresses can be IPv4 addresses while outer addresses are IPv6 addresses, and vice versa.

Since inner addresses are not allowed to change for a given SA, they do not need to be transmitted in each ESP packet. They are exchanged only during the SA establishment phase. Therefore, the format of ESP packets in BEET mode is the same as in transport mode. The encrypted payload data contain only the IP payload, based on inner addresses. The IP header of the generated packet is however created with the outer addresses.

The encapsulation of IP packets with ESP in BEET mode is processed as follows :

- encryption of the IP payload based on inner addresses,
- addition of the ESP header and trailer, before and after the encrypted data,
- computation of the authentication field, placed at the end of the packet,
- generation of a new IP header based on outer addresses and placed before the ESP header.

The decapsulation of ESP packets in BEET mode is processed as follows :

- verification of the authenticity of the packet,
- decryption of the payload data field,
- generation of a new IP header based on inner addresses and appended to the decrypted data.

The original and encrypted versions of an IP packet containing TCP data is described in Figure 2.9.

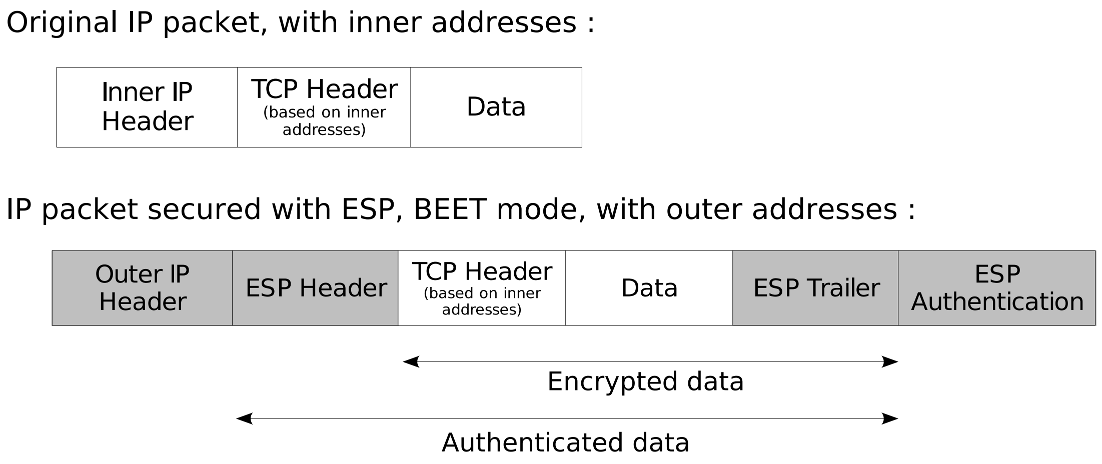
Original IP packet, with inner addresses :
Inner IP Header
TCP Header (based on inner addresses)
Data
IP packet secured with ESP, BEET mode, with outer addresses :
Outer IP Header
ESP Header
TCP Header (based on inner addresses)
Data
ESP Trailer
ESP Authentication
Encrypted data
Authenticated data

Figure 2.9: A TCP-IP packet and its corresponding secured version in ESP BEET mode.

This new mode for ESP has been presented because it is adequate for HIP communications. As it is discussed in further sections, the BEET mode is perfectly adapted to the HIP model, in particular with the possibility to use HITs as inner addresses in HIP SAs.

2.3 Network Address Translators

The concept of Network Address Translation has been presented in Chapter 1. The corresponding devices, called Network Address Translators (NATs), have been originally introduced to deal with the lack of IPv4 addresses. They allow multiple hosts located on a private network to access the Internet with the same public IP address. Traditional NATs are described in [8].

NAT functioning relies on a mapping table in which the NAT registers for each ongoing communication, the relation between the address of the private host and its own corresponding external address. The mapping may be based only on IP addresses, but may also include transport layer port numbers, in particular for UDP and TCP communications. Thus two types of network address translation are distinguished. First the *basic* or *static* network address translation, where only the IP addresses of the packet going through a NAT are translated. And second, the network address and port translation where both IP addresses and port numbers may be modified by the NAT. In this second case, the devices are sometimes referred to as Network Address and Port Translators (NAPTs).

The type of entry in the mapping table of a NAT (or NAPT) and the rules used to establish a relation between a private address and an external address depend on the NAT itself. Indeed several types of NAT exist, and the main categories of them are described in the following section. Since most of the NATs are able to process port translation, we will mainly focus on NAPTs in the following sections. The term *NAT* designates however both basic NATs and NAPTs in the following sections, unless an explicit distinction between them is required.

2.3.1 Different Types of NATs

NATs can be separated into four main categories.

Full Cone NATs map all the requests from a given pair composed of an internal IP address and a port number, to a single pair composed of an external IP address and a port number. The internal and external port numbers may be different. All the traffic coming from the public realm and destined to the specific external IP address/port number pair is translated and forwarded to the corresponding private host. The NATs do not apply any restriction on the source address of incoming traffic.

Restricted Cone NATs perform the same translations as Full cone NATs. An internal IP address and port pair is mapped to a single external IP address and port pair. However, Restricted Cone NATs apply some restriction to the incoming traffic. An external host with a given IP address may send some data to an internal host only if the internal host had sent some data to the specific IP address of the external host beforehand.

Port Restricted Cone NATs act like Restricted Cone NATs with a source restriction extended to the port numbers. An external host may send some data with given source IP address IP_a and source port number P_a to an internal host, only if the internal host had previously sent some data to the IP address IP_a and port P_a.

Symmetric NATs establish mapping based on the internal source IP address and port number and on the external destination IP address and port number. Thus a traffic coming from a given pair of internal IP address and port, and destined to a given pair of IP address and port is mapped to a specific pair of external IP address and port. Another traffic with same source IP address and port but destined to a different pair of IP address and port, would be mapped to a different pair of external IP address and port. Furthermore, only the destination host of the traffic is allowed to reply to the internal host and it must use the same source and destination pairs as the incoming traffic but with inversed roles.

To explain more clearly the differences between the various types of NATs, some examples of their behavior are presented in Appendix A. These examples describe which kinds of traffic are either properly translated and forwarded, or rejected, according to the type of the NAT.

In addition, for all types of NAT, incoming data flows which do not match any entry of the mapping table are rejected by the NATs. This implies that communications with a host located behind a NAT must always be initiated by the internal host itself.

Furthermore, when an unsupported transport layer protocol is used, NATs will discard the corresponding traffic. In particular NATs that perform port translation may translate only traffics providing transport layer port numbers, thus basically UDP and TCP flows, and prevent all other forms of traffic. In such cases, NATs become really restricting devices. This point is developed in the following section.

2.3.2 NAT Traversal Issue with HIP

In HIP communications, the two main phases have to be distinguished in the discussion about NAT traversal. Since the HIP base exchange and the secured user data transfer generate two different kinds of traffic, their behaviors regarding NATs must be considered separately.

First the HIP base exchange is based on specific HIP packets that have been described in Section 2.1.2.1. These packets do not contain transport layer port numbers. Therefore, a HIP traffic may encounter difficulties in NAT traversal. NAPTs that require port numbers to establish a translation will drop HIP traffic. Furthermore, HIP traffic may even be blocked by less restrictive NATs. Indeed HIP is still under development and is consequently very likely to be an unsupported protocol for most of current NATs. In such conditions, NATs which do not know how to handle HIP packets, will simply discard them.

NAT traversal for HIP packets constitutes a major issue, because most current NATs will not translate and forward them. Moreover, even NATs that would accept HIP traffic, would also introduce another problem for HIP packets. The problem concerns transport layer checksums. HIP packets contain a checksum in their header that is calculated with the IP addresses used by the sending host. Since NATs modify IP headers while translating packets, the transport layer checksums become invalid, unless the NATs correct them before forwarding the translated packets. This is the case for TCP and UDP flows. TCP and UDP checksums are usually recomputed by NATs before the packets are forwarded. But once again, the majority of current NATs do not have any HIP support and consequently no checksum correction procedure will be performed for HIP packets. Eventually HIP packets with wrong checksums are dropped by HIP-aware hosts (either the recipient of the packet itself, or another intermediary HIP-aware device).

The second phase of HIP communications, the secured data transfer, is also subject to NAT traversal issues. The main security protocol used for this second phase is IPsec ESP and this protocol encounters similar difficulties for NAT traversal as the HIP protocol. The NAT traversal issues for ESP are described in [9].

HIP communications are currently based on specifications that suppose direct connection between the end-hosts, without middleboxes modifying their traffic on the path between them. Therefore, HIP does not have any support for NAT traversal in its current status. Two solutions are possible to solve the NAT traversal issue : adapt the NATs to become HIP-aware and handle HIP traffic properly, or improve the protocol itself to include a NAT traversal support. Since the amount of NAT devices currently used in the Internet is really high and since it is not reasonably possible to expect all of them to be updated to become HIP-aware, the first solution does not appear as feasible. Consequently a support for NAT traversal in HIP specifications is required. This is the purpose of the present thesis, which provides an extension of HIP to enable NAT traversal.

Chapter 3

NAT Traversal Extension for HIP

In this chapter, the extension of HIP specifications to provide NAT traversal support is described. Section 3.1 exposes the general principles of NAT traversal for HIP communications. Sections 3.2 and 3.3 describe more specific details for the case where respectively the initiator or the responder of a HIP communication is located behind a NAT.

3.1 NAT Traversal : General Principles

In this section, the general priniciples of NAT traversal mechanisms are presented. The NAT detection mechanism required to determine whether or not NAT-traversal measures must be applied, is discussed in Section 3.1.1. Then, actual NAT-traversal mechanisms are presented in Section 3.1.2.

3.1.1 NAT Detection

An important phase in a NAT traversal support extension is to determine the presence or absence of NAT between the communicating nodes, in order to know whether specific measures have to be taken or not. NAT traversal is a major issue for many protocols and applications, so that various NAT detection systems have been developed to inform a host about its status regarding NATs. Since protocols have already been defined to detect the presence and the type of possible NAT(s) between a host and a given network, the NAT traversal extension for HIP will rely on such protocols to determine whether or not a HIP node is located behind a NAT. Consequently, no specific procedure will be developed in HIP. The results of an external NAT detection mechanism are directly used to adapt HIP nodes behavior towards possible NATs.

A suggested NAT detection protocol is called Simple Traversal of UDP over NATs (STUN) and is defined in [10]. STUN allows a host to determine whether it is located behind one or several

NATs and in such case, the type of the encountered NAT(s) as defined in Section 2.3.1.

STUN is a client-server protocol. A STUN client sends a request to a STUN server located on a public network and can deduce from the replies of the server if it is located behind a NAT or directly connected to the public network. If the STUN client is located on a private network protected by a NAT, the server responses also indicate to the client which public IP address the NAT provided to it.

The principle of STUN consists in an exchange of UDP packets between the STUN client and the STUN server. The client includes some parameters in the requests to indicate to the server which kind of IP address or port number it must use to send the replies. A STUN server owns typically two different IP addresses and thus can use either the IP address to which the request was destined or the second IP address to send its answer back. The UDP port numbers used in the responses of the STUN server can also be identical or different from the UDP port numbers used in the received request packets and which are the port numbers provided by the last encountered NAT between the client and the server. In that way, the STUN client can deduce from the answers it receives or does not receive from the server, the type of NATs that separate it from the public network, and the kind of restrictions these NAT apply to the incoming and outgoing traffics.

Thus in the next sections, we assume that a HIP implementation is coupled to a NAT detection mechanism (such as STUN) and has direct access to the information that such a system can provide. The following specifications for HIP are consequently based on the assumption that a host can always know whether it is located behind a NAT or not, and if necessary which type of NAT modifies its data flows.

3.1.2 NAT Traversal for HIP Communications

The objective of the present extension of HIP specifications consists in solving the issues due to the presence of NATs and pointed out in Section 2.3.2. Thus the new specifications must first prevent all kind of HIP traffic packets from being dropped by encountered NATs, and second prevent these packets from being invalidated by IP header modifications.

The common solution to the first problem consists in encapsulating every packet of the HIP communication in UDP packets, so that the transport layer protocol observed by NATs is a well known protocol, properly translated and forwarded. This solution is especially the procedure recommended for the NAT traversal support of common IPsec ESP traffic. The UDP-encapsulation of common ESP packets is described in [11].

Thus the present thesis provides specifications for the UDP-encapsulation of both HIP packets and ESP packets used in HIP communications. To prevent the packet invalidation issue, a specific handling of transport layer checksums is also required and described in the further sections. Eventually, the BEET mode for ESP is introduced in the secured data transfer of HIP communications. The reasons for this choice is explained in Section 3.1.2.2.

3.1.2.1 UDP Encapsulation of HIP Packets

The UDP encapsulation consists in inserting a UDP header between the IP header of a packet and the original payload of the IP packet. The UDP header format is defined in [4] and described in Figure 3.1.

0 1 2 3 4 5 6 7 **8** 9 10 11 12 13 14 15	**16** 17 18 19 20 21 22 23 **24** 25 26 27 28 29 30 31
Source Port Number	**Destination Port Number**
Length	**Checksum**

Figure 3.1: UDP header format.

The UDP header is composed of two port numbers used to determine to which services the packets are related, the length of the packet and a checksum. Since the HIP packet header contains already the length of the packet and a checksum, a possible solution could be to slightly modify the HIP header by reordering the fields and adding two port numbers, so that the HIP header appears like a UDP header. This solution has the advantage to reduce the UDP encapsulation overhead to its minimum. It presents however several issues. First it requires the use of two distinct HIP headers which is not a suitable solution. But moreover it would divert the use of UDP headers and mix UDP and HIP protocols. From a protocol design point of view, this appears as a bad and unclear solution.

Thus the proper way to perform UDP encapsulation consists in including the entire HIP packet in the UDP payload. This process may duplicate some already present information in the HIP header, but it is clearer and cleaner than modifying HIP headers to fake UDP headers.

To complete the UDP encapsulation of HIP packets, the mechanism must include a process to handle HIP header checksums. Since HIP checksums are based on the IP addresses used by the sending host, they may be invalidated by the IP header modifications performed by NATs. Furthermore, the UDP header added to the HIP packet introduces a new checksum which will be maintained valid by the possible encountered NATs. This checksum is sufficient to verify the integrity of the packet. Consequently the HIP header checksum is not useful anymore and can be set up to zero.

The UDP encapsulation and decapsulation procedures for HIP packets are straightforward. When sending a UDP-encapsulated HIP packet, a HIP implementation that supports the proposed NAT traversal extension must zero the HIP header checksum before computing the UDP header checksum. Then it introduces a properly formatted UDP header before the HIP header. The IP header of the packet will then contain the UDP protocol number instead of the HIP protocol number in its transport protocol field, and the total length field needs also to be updated.

The UDP encapsulation procedure for a HIP packet is described in Figure 3.2.

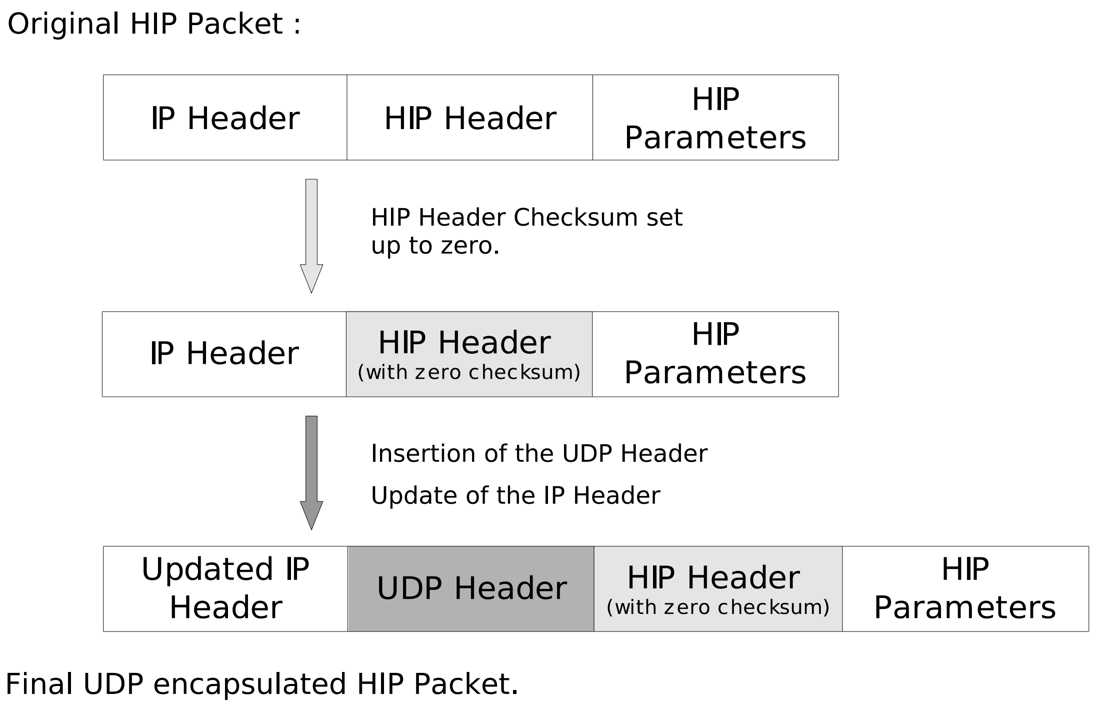
Original HIP Packet :
IP Header
HIP Header
HIP Parameters
HIP Header Checksum set up to zero.
IP Header
HIP Header (with zero checksum)
HIP Parameters
Insertion of the UDP Header
Update of the IP Header
Updated IP Header
UDP Header
HIP Header (with zero checksum)
HIP Parameters
Final UDP encapsulated HIP Packet.

Figure 3.2: UDP encapsulation procedure for a HIP packet.

The decapsulation procedure for UDP-encapsulated HIP packets consists first in the verification of the correctness of UDP header checksum exclusively. The checksum included in the HIP header must not be taken into account. Then the UDP header is discarded and the IP packet containing only the HIP packet is reconstituted and transmitted to the common HIP processing mechanism.

3.1.2.2 Secured Data Transfer with BEET-Mode ESP

One of the main decision of the present extension for HIP communications is to replace ESP transport-mode security associations with ESP BEET-mode SAs for the secured user data transfer.

The main advantage of the IPsec ESP BEET mode over the common ESP transport mode is the explicit definition of inner and outer addresses. The inner addresses of BEET associations are used for processing at all layers of the stack, from the IPsec layer to the application layer, whereas the outer addresses are only used to transmit the final packets on the wire. This explicit distinction between addresses matches perfectly the IP address role separation that HIP wants to achieve with the introduction of new identifiers. Indeed, HITs can be used as IPv6 inner addresses because they all have a role of static identifier, and the hosts current IP addresses are defined as the outer addresses of the BEET associations and take the role of locators. The end-host IP addresses may be modified when a mobile host changes its location. The BEET outer addresses can then be updated accordingly, since they are allowed to be modified during the lifetime of the corresponding BEET associations.

Furthermore, using BEET associations is helpful regarding NAT traversal issues and is a way to apply explicitly the measures that the current specifications of HIP assume in a more partial and implicit way. The transmission of data encrypted with ESP in transport mode is indeed subject to the checksum invalidation issue. If the secured traffic traverses NATs, the IP addresses contained in the IP header of the packet received by an end-host are likely to be different from the addresses used to compute the possible transport layer checksums contained in the encrypted data. Thus in transport mode, the receiver of the packet does not have the proper information to verify the correctness of the inner checksums and will then consider the packet invalid. To avoid this problem, HIP specifications in [13] already indicate to modify the TCP and UDP checksums of the user data and recompute them using the HITs as IP addresses in an IPv6 pseudo-header. Thus HIP is already designed to perform some tunnelling for the secured transmission of the user TCP and UDP flows.

Therefore, the use of the BEET mode instead of the transport mode renders this tunnelling explicit for TCP and UDP flows, and also makes it available for other transport layer protocols. The tunnelling then solves the checksum invalidation issue for the encrypted user data, because the checksums will be based on the inner addresses which are known by both end-hosts and do not change during the communication.

This solution could also be achieved with the common tunnel mode for ESP. However, the BEET mode has two advantages over the common tunnel mode in a HIP communication context. First the BEET mode avoids any additional overhead, because the ESP packet structure is similar to the transport mode. And second, no additional exchange is required to set up the SAs, because the BEET mode allows the use of HITs as inner addresses and HITs are already known and transmitted during the HIP base exchange.

Nonetheless, although the tunnelling property of the BEET mode avoids transport layer checksum invalidation, the ESP BEET mode traffic remains subject to the common difficulties of NAT traversal. The UDP-encapsulation of ESP flows recommended and described in [11] is therefore still required. However, the encapsulation and decapsulation procedures for ESP BEET mode packets has not been defined yet. These procedures are thus described in the next section.

3.1.2.3 UDP Encapsulation and Decapsulation Procedures for BEET-Mode ESP Packets

The UDP encapsulation and decapsulation procedures for the common transport and tunnel modes of ESP are defined in [11]. The procedures for BEET-mode ESP are relatively similar.

The UDP encapsulation is processed according to the following steps :

- The original packet targeted for UDP BEET-mode encapsulation is an IPv6 packet containing the HITs of the communicating end-hosts as source and destination IP addresses. If any transport layer checksum is present in the payload data, it must have been computed with the HITs indicated in the IPv6 header of the packet. The packet must then undergo the BEET-mode ESP cryptographic processing using the encryption protocol and parameters indicated in the corresponding BEET SA, as defined in [20].

- The resulting BEET-mode ESP packet is then encapsulated in UDP. For this purpose, a common UDP header is inserted between the existing IPv6 and ESP headers. The UDP checksum must be calculated based on a pseudo IP header containing the outer addresses specified in the corresponding BEET SA. The source and destination port numbers of the UDP header must also be set up to the values that have to be specified in the SA.

- The resulting UDP packet must then undergo the final step of the BEET-mode encryption procedure, the IP header processing. During this step, the inner IPv6 header based on HITs is replaced by a new IP header based on the outer addresses specified in the BEET SA. The resulting packet can then be sent on the wire.

Figure 3.3 describes the encapsulation procedure for a packet containing a TCP segment.

When a UDP-encapsulated BEET-mode ESP packet is received, the following decapsulation procedure is performed :

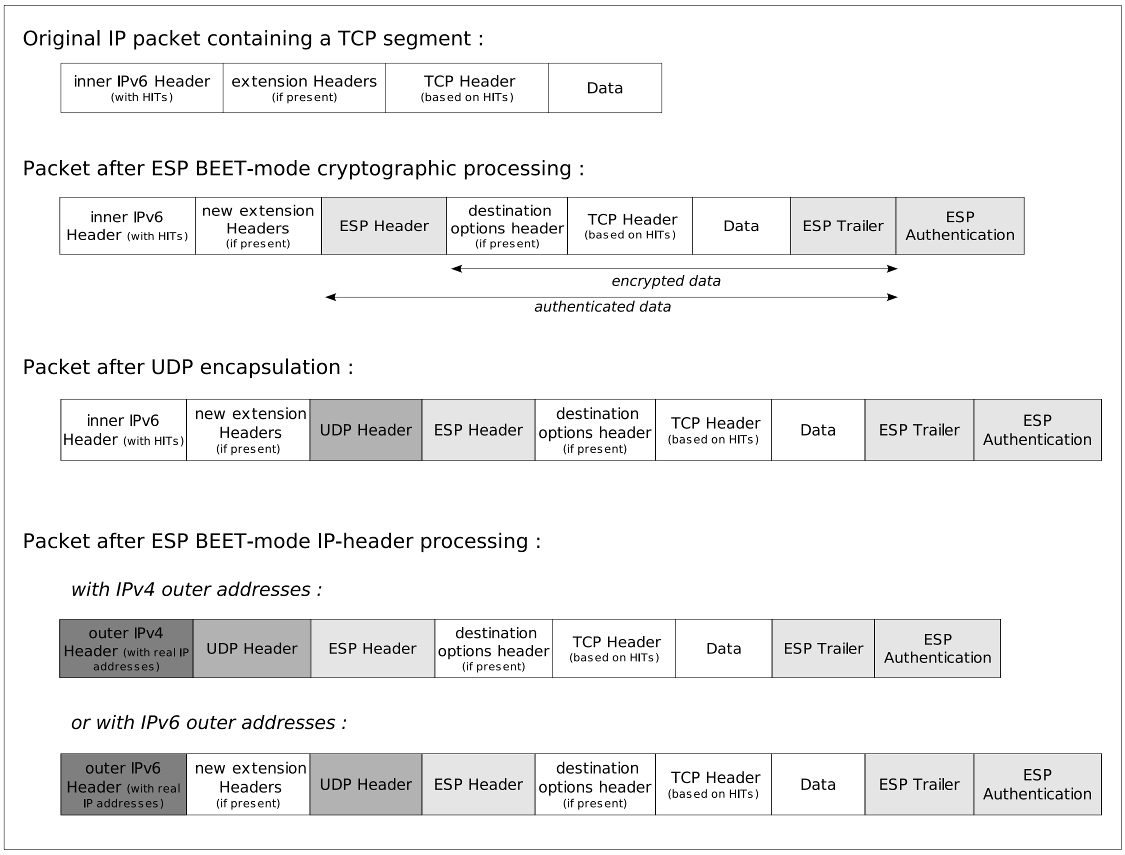
Original IP packet containing a TCP segment :
inner IPv6 Header (with HITs)
extension Headers (if present)
TCP Header (based on HITs)
Data
Packet after ESP BEET-mode cryptographic processing :
inner IPv6 Header (with HITs)
new extension Headers (if present)
ESP Header
destination options header (if present)
TCP Header (based on HITs)
Data
ESP Trailer
ESP Authentication
encrypted data
authenticated data
Packet after UDP encapsulation :
UDP Header
Packet after ESP BEET-mode IP-header processing :
with IPv4 outer addresses :
outer IPv4 Header (with real IP addresses)
or with IPv6 outer addresses :
outer IPv6 Header (with real IP addresses)

Figure 3.3: UDP encapsulation procedure for a TCP packet secured with ESP BEET-mode.

- The UDP header checksum must be verified. If the verification fails, the packet is dropped. Otherwise the Security Association to which the packet is related, is determined in function of the destination IP address of the packet, the security protocol used (ESP) and the SPI indicated in the ESP header. Then according to the authentication algorithm specified in the SA, the integrity of the ESP packet is checked.

- After the packet integrity has been validated, the payload data of the ESP packet is decrypted based on the encryption protocol and parameters indicated in the BEET SA. The outer IP header, the UDP header and the ESP header and trailer are discarded.

- A new IPv6 header is generated based on the inner addresses which are the end-hosts HITs. This header is then prepended to the decrypted data and the resulting packet is transmitted to the upper layers of the network stack.

3.1.2.4 Keep-Alive Mechanism

Usually the bindings registered in the mapping tables of NATs have a limited lifetime when they are not used during a certain time. NATs indeed time the established bindings out if they have not used them to relay traffic for a given period of time, varying from one NAT to another. To prevent NAT bindings that support the traversal of UDP-encapsulated HIP traffic from timing out during times when there is no control (HIP) or data (ESP) traffic, HIP hosts have to send periodic keep-alive messages.

Such keep-alive mechanisms concern both HIP and ESP traffics. If the UDP port numbers used by these two types of traffic are different, keep-alive messages have to be generated separately for each flow, because they would correspond to two different mappings in the encountered NATs mapping tables.

Typically NATs only act, for security reasons, on keep-alive messages received from hosts located in the private network that they connect to the public Internet. Consequently those hosts located behind NATs have to send periodic keep-alive messages for the control and data channels of all their established HIP associations, if the respective channel has been idle for a specific period of time. Keep-alive intervals for HIP control and data channels are thus two new parameters that have to be included in the configuration of HIP associations.

Keep-alive messages can be minimal UDP packets created with the same source and destination IP addresses and port numbers as the ones used for the corresponding UDP-encapsulated HIP or ESP flow. The keep-alive message suggested in [11] for common UDP-encapsulated ESP flows is a basic UDP packet which carries a payload constituted of a single octet. The single-octet payload is set up to the value 0xFF. Such a packet is illustrated in Figure 3.4.

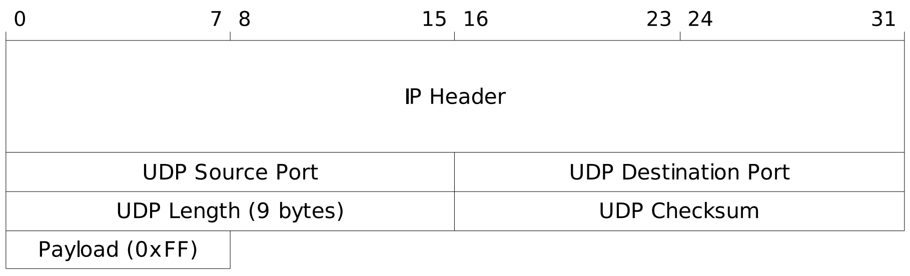

Figure 3.4: Common keep-alive packet for UDP-encapsulated ESP flows, constituted of a UDP packet with a one-byte payload.

Usually keep-alive messages are simply discarded by their recipient. However, the present extension of HIP uses keep-alive packets for another purpose than just keeping the NAT bindings alive. Indeed, keep-alive packets are also used in special cases to determine the public IP addresses and port numbers used by NATs to forward the HIP and ESP traffics. In such conditions, a basic UDP packet with a one-byte payload is not sufficient and more elaborated keep-alive packets are required.

One of the particuliar cases consists in the reactivation of ESP channels after a mobile host has changed its location. This case, which is described in Section 3.2.5, may require the sending of a specific packet that enables the static host to determine which ESP channel has been reactivated and to which SA it is related. For this purpose, the specific packet must contain an ESP header corresponding to the reactivated ESP channel. Since the encrypted data are not important in this context, a single-octet payload, encrypted with ESP, would be sufficient. The suggested packet is consequently a normal UDP-encapsulated ESP packet, carrying an encrypted one-byte payload of value 0xFF. This packet is described in Figure 3.5.

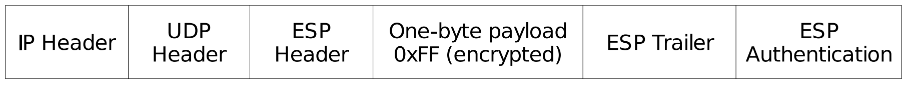

Figure 3.5: Specific keep-alive packet for UDP-encapsulated ESP flows, suggested for the reactivation of ESP channels.

3.1.2.5 UDP Channels and Port Numbers

The present NAT traversal extension for HIP communications defines the UDP-encapsulation of both HIP control and ESP data traffics. An important issue to solve is whether to use the same UDP channel for both types of traffic or transmit them on two separated channels. A related issue is also to determine the single or multiple UDP port numbers to use for the encapsulated flows. During the development phase of HIP, port numbers can be chosen freely in the range of *Dynamic and/or Private Ports* (from 49152 through 65535). However, the official definitive port numbers will have to be provided and registered by the Internet Assigned Numbers Authority (IANA).

Transmitting the encapsulated HIP and ESP flows on the same UDP channel or transmitting them on two different channels present both advantages and detriments.

The main advantage of using a single channel for HIP and ESP traffic is that it facilitates the UDP channels management. In particular in the mobility scenarios, only one UDP channel per HIP association has to be taken care of. For example, if a mobile host new location is protected by a NAT, the activation of a new UDP tunnel and the suppression of the possible former one has to be performed only once for each HIP association. If HIP and ESP traffics are encapsulated in two different UDP flows, the procedure has to be executed separately for each protocol. Furthermore, using a single channel requires the introduction of only one keep-alive system, which will operate for both HIP and ESP traffics.

However, using a single UDP channel for both HIP and ESP traffics also implies the definition of a multiplexing and demultiplexing mechanism to allow end-hosts to distinguish between the two types of encapsulated traffic. This would therefore introduce an extra complexity in the protocol specifications.

Moreover, some UDP port numbers have already been defined in [11] for the common UDP-encapsulated ESP traffic. To facilitate the implementation and integration of HIP in current platforms, using the same UDP channels for HIP communication ESP traffic as common ESP traffic would be a good option. The current specifications of UDP-encapsulated IPsec ESP assign the port number 4500 to UDP-encapsulated ESP transport-mode and tunnel-mode packets. This port number is also shared with the Internet Key Exchange (IKE) protocol [12]. The distinction between UDP-encapsulated IKE and IPsec ESP packets is done based on the SPI field of the ESP headers. This port number has been registered for IKE and IPsec ESP traffic only. Consequently UDP-encapsulated HIP packets will not be authorized to use this port number.

To be able to join the ESP traffic of HIP communications to the common ESP traffic, and also to avoid the introduction of an additional multiplexing mechanism, the present extension for HIP is based on the use of two separated UDP channels for UDP-encapsulated HIP control and ESP data flows.

This choice implies the use of two different port numbers and the separated management of HIP and ESP flows. In particular two separated keep-alive systems are required and updating procedures in mobility scenarios have to be defined explicitly for both HIP and ESP protocols.

The port number chosen for UDP-encapuslated HIP traffic is 50500. The port number for UDP-encapsulated ESP traffic can not be set up to the encapsulated IKE/IPsec ESP port number (4500) because the BEET mode, still under development, has not been recognized as regular IPsec ESP mode yet. Thus another experimental port is currently defined for UDP-encapsulated BEET-mode ESP packets in HIP communications. The chosen port number is 54500.

HIP implementations that support NAT traversal, must therefore listen for incoming traffic on UDP ports 50500 and 54500.

Since HIP is still under development, it has to be noticed that the choice of separated UDP channels is not a definitive decision and may be modified in future versions of the specifications. In particular the choice of a unique port number for both HIP and ESP flows may finally be adopted, based on the feedback of experimental implementations, which state that the avoided additional complexity of multiplexing mechanism is replaced by an extra complexity in the management of UDP channels in mobility scenarios (see Section 3.2.5).

3.2 First Case : Initiator Behind a NAT

The general principles of the NAT traversal extension have been described in the previous sections. Nonetheless, precise specifications of HIP behavior still have to be defined to determine in which cases the NAT traversal measures have to be applied, which operations have to be performed and in particular how to handle mobility scenarios. To achieve this goal, two main cases have to be distinguished. Since NATs do not have a symmetric behavior towards the hosts located on their private network and the hosts located on the public network, the NAT traversal extension differentiates the case where only the initiator of a HIP communication may be located in a private network protected by a NAT, from the second case where the responder of a HIP communication may also be located behind a NAT. The former case is handled in the current section, and the second case is treated in Section 3.3.

In the present section, the responder of a HIP communication is thus assumed to be in the global Internet. If the initiator of the communication is also located in the global Internet, then the common HIP specifications can be applied. Otherwise, if the initiator detects the presence of a NAT modifying its traffic, it has to apply the NAT traversal measures introduced in the previous section and more precisely described in the following sections.

3.2.1 HIP Control Traffic

When the initiator of a HIP communication has detected the presence of a NAT from any of the types described in Section 2.3.1, it must apply NAT traversal measures for the HIP control traffic.

Thus HIP packets and in particular HIP base exchange packets have to be encapsulated in UDP as described in Section 3.1.2.1. The initiator must use the port number defined for UDP-encapsulated packets (50500) as destination port number for all the HIP control packets that it sends. The source port number may be set up to 50500 as well, but the initiator has also the possibility to choose a random, unoccupied source port. Since the source port number is very likely to be modified by the NAT translation, the choice of the initiator is not an important matter for the continuation of the base exchange. Moreover, using a random source port number instead of a static one renders possible to have multiple clients behind a NAT middlebox that performs only address translation and no port translation. Nevertheless, if the initiator uses a random source port number, it must then listen for and accept all HIP control packets arriving on this port until the corresponding HIP association is torn down. A random source port number must be in the range of the dynamic and private ports (49152-65535).

The responder of a UDP-encapsulated HIP base exchange must also apply the NAT traversal measures and encapsulate in UDP all the HIP control traffic of the corresponding association. The responder must use the port number 50500 as source port number for all UDP-encapsulated control packets it sends, because 50500 was the number of the port on which it received prior packets from the initiator. As a source IP address, the responder must use the IP address on which prior packets from the initiator arrived. Similarly it must use the source IP address and source UDP port of prior packets from the initiator, as destination IP address and destination UDP port.

Whether or not HIP control packets are UDP-encapsulated does not affect the HIP state machine. HIP implementations applying the present specifications must process all UDP-encapsulated control messages equivalently to unencapsulated control messages. The single exception to this rule concerns IPsec ESP security associations. If the HIP base exchange is UDP-encapsulated, the secured user data transfer is based on a UDP-encapsulated BEET-mode ESP association (see Section 3.2.3).

Furthermore, in the case of UDP-encapsulated HIP exchanges, both initiator and responder must register the source and destination IP addresses and port numbers they have to use to transmit

HIP packets. These IP addresses and port numbers must be stored independently for each HIP association.

3.2.2 HIP Base Exchange

As mentioned in the previous section, when the initiator of a HIP communication is located behind a NAT, it triggers a UDP-encapsulated HIP base exchange. Figure 3.6 describes the HIP base exchange between an initiator with the private IP address IP_i and a responder with the public IP address IP_r. The initiator is located behind a NAT, which has the private IP address IP_{priv} and the public IP address IP_{pub}.

The initiator begins the base exchange by sending a UDP-encapsulated I1 packet to the responder. According to the rules previously specified, the source IP address of this I1 packet is IP_i and its UDP source port number is p_i. It is addressed to IP_r on port 50500. The NAT forwards the I1 packet but substitutes the source IP_i with its own public IP address IP_{pub} and substitutes the UDP source port p_i with $p_{nat.1}$, which will usually be different from p_i.

When the responder receives the UDP-encapsulated I1 packet on the UDP port 50500, it processes it according to the common HIP specifications. If it replies with an R1 packet, this packet uses the destination IP address and UDP port from the previous I1 packet (i.e. IP_r and 50500), as source IP address and UDP source port number. And similarly, the destination IP address and UDP port of the R1 packet are set up to the source IP address and UDP port of the I1 packet, i.e. IP_{pub} and $p_{nat.1}$. The NAT then substitutes the destination of the R1 packet, replacing the pair $IP_{pub} : p_{nat.1}$ with $IP_i : p_i$.

When the initiator receives the UDP-encapsulated R1 packet from the responder, it processes the packet according to the common HIP specifications. When the initiator responds with a UDP-encapsulated I2 packet, it uses the same source and destination IP addresses and the same source and destination UDP ports that it used previously for sending the corresponding I1 packet. The I2 packet is thus addressed to the destination $IP_i : p_i$, with the source $IP_r : 50500$. The NAT again substitutes the source information, replacing it with $IP_{pub} : p_{nat.2}$.

When the responder receives the UDP-encapsulated I2 packet destined to the UDP port 50500, it uses the UDP source port contained in this packet for further HIP communications with the initiator. It then processes the I2 packet according to HIP specifications. When it responds with an R2 message, it UDP-encapsulates this message, using the UDP source port of the I2 packet as the destination UDP port, and sends it to the source IP address of the I2 packet. The responder thus addresses the R2 packet to the destination $IP_{pub} : p_{nat.2}$, and sets up the source as $IP_r : 50500$.

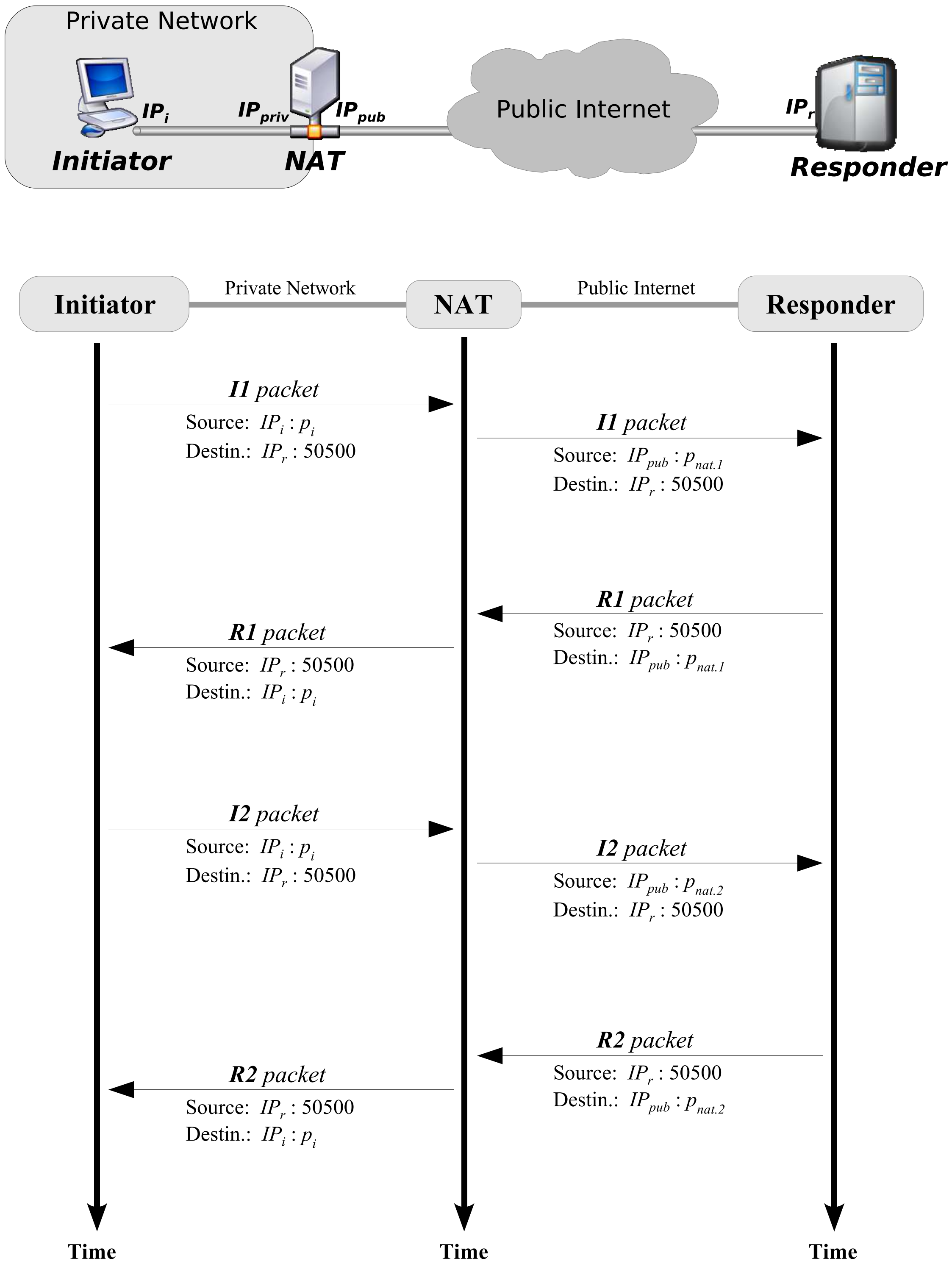

Figure 3.6: UDP-encapsulated HIP Base Exchange between an initiator behind a NAT and a responder in the public Internet.

The NAT will then again replace the destination information in the packet with $IP_i : p_i$.

Usually, the I1-R1 and I2-R2 exchanges occur fast enough for the NAT binding to not time out. This means that the NAT uses the mapping established during the I1-R1 exchange to translate the I2-R2 exchange. Thus the ports $p_{nat.1}$ and $p_{nat.2}$ will be identical. However, the mechanism can handle the case where the NAT state times out between the two exchanges and the I1 and I2 arrive from different UDP source ports and/or IP addresses.

3.2.3 Security Associations

The next phase of the HIP communication is the secured user data transfer. The corresponding exchanges are secured with the ESP protocol and thus based on the ESP security associations defined during the HIP base exchange. When a HIP base exchange is UDP-encapsulated, the HIP nodes must define UDP-encapsulated BEET-mode SAs instead of the common transport-mode SAs. In such case, no additional parameter has to be added to the HIP base exchange packets. UDP-encapsulated BEET-mode SAs require more parameters to be properly defined, in particular inner addresses, but all necessary parameters are already included in the usual HIP base exchange packets.

The main differences between the usual transport SAs and the present BEET-mode SAs are the different ESP mode used, the activation of the UDP-encapsulation and the specific definitions of IP addresses and UDP port numbers. The management of encryption and authentication protocols and SPIs occurs in the same way as in the usual case of transport-mode SAs (described in [15]).

Each HIP node must define two SAs for each HIP association in which it is implicated. One SA is destined to the protection of outgoing data. The second SA is destined to the protection of incoming data coming from the peer. The inbound SA of a HIP node is symmetrically defined to the outbound SA of its peer, and vice versa.

The initiator of the base exchange must also initiate the BEET-mode ESP exchange, by either sending a UDP-encapsulated ESP data packet, or a specific keep-alive packet (as defined at the end of Section 3.1.2.4) if it has no data to send. This must occur immediately after the base exchange has succeeded. At this moment the initiator of the base exchange has indeed completed the definition of its SAs and can therefore send secured data to the responder. This is however not the case for the responder. Since SAs must contain the IP addresses and UDP port numbers to use, the responder must wait for the first UDP-encapsulated ESP packet coming from the initiator to know which are the exact IP address and UDP port number used by the last encountered NAT to forward the ESP traffic. The HIP and ESP traffic are transmitted in two different UDP channels and are consequenlty related to different bindings in the NATs. For a given HIP association, the UDP-encapsulated HIP and ESP flows are likely to arrive to the responder from two different IP-address/UDP-port pairs. Therefore, the first correct UDP-encapsulated ESP packet arriving after the base exchange indicates to the responder how to define the source IP-address/UDP-port pair of the inbound SA, and the destination pair of the outbound SA.

The initiator of the base exchange must use the general port number of UDP-encapsulated BEET-mode ESP (54500) as destination port for all the UDP-encapsulated ESP packets it sends. In the same way as UDP-encapsulated HIP traffic, the initiator may use the same port number as source port, but it can also choose a random, unoccupied port number in the range 49152-65535. If it uses a random port number, it then has to listen for and accept UDP-encapsulated ESP packets arriving on this port, and this, until the corresponding HIP association is torn down.

The responder, which is listening for packets arriving on port 54500, must use this port number as source port for all the UDP-encapsulated ESP packets it sends back to the initiator. And it must use the source port number of prior UDP-encapsulated ESP packets from the initiator, as destination port number fot the packets it sends.

The two following sub-sections describes more precisely the parameters of respectively the initiator SAs and the responder SAs.

3.2.3.1 Initiator Security Associations

The initiator behind at least one NAT defines its SAs during the HIP base exchange. It must define the parameters of its outbound SA as follows :

IPsec ESP mode : BEET mode with UDP encapsulation,

SPI : the value the responder chose during the base exchange,

Inner source address : the local HIT used during the base exchange,

Inner destination address : the responder HIT used during the base exchange,

Outer source address : the local IP address from which the base exchange packets were transmitted,

Outer destination address : the responder IP address to which the base exchange packets were transmitted,

UDP source port : the port 54500 or another unoccupied port randomly chosen by the initiator itself,

UDP destination port : the port 54500.

Similarly, the initiator must define its inbound SA as follows :

IPsec ESP mode : BEET mode with UDP encapsulation,

SPI : the value the initiator chose itself during the base exchange for the rececption of the ESP packets coming from the responder,

Inner source address : the responder HIT used during the base exchange,

Inner destination address : the local HIT used during the base exchange,

Outer source address : the responder IP address to which the base exchange packets were transmitted,

Outer destination address : the local IP address from which the base exchange packets were transmitted,

UDP source port : the port 54500,

UDP destination port : the same port as the UDP source port the initiator defined in its outbound SA.

3.2.3.2 Responder Security Associations

The responder must define its SAs after the reception and correct processing of the first UDP-encapsulated ESP packet coming from the initiator. The responder must use the SPI value it chose itself during the base exchange, to process the incoming ESP packet. Since this packet may have traversed several NATs, the source IP address and port number it contains are likely to be different from the values used at the initiator.

The responder must set up its outbound SA as follow :

IPsec ESP mode : BEET mode with UDP encapsulation,

SPI : the value the initiator chose during the base exchange,

Inner source address : the local HIT used during the base exchange,

Inner destination address : the initiator HIT used during the base exchange,

Outer source address : the local IP address from which the base exchange packets were transmitted,

Outer destination address : the source IP address of the first correctly processed UDP-encapsulated ESP packet received from the initiator,

UDP source port : the port 54500,

UDP destination port : the UDP source port of the first correctly processed UDP-encapsulated ESP packet received from the initiator.

The responder similarly defines its inbound SA as follows :

IPsec ESP mode : BEET mode with UDP encapsulation,

SPI : the value the responder chose itself during the base exchange,

Inner source address : the initiator HIT used during the base exchange,

Inner destination address : the local HIT used during the base exchange,

Outer source address : the source IP address of the first correctly processed UDP-encapsulated ESP packet received from the initiator,

Outer destination address : the local IP address from which the base exchange packets were transmitted,

UDP source port : the UDP source port of the first correctly processed UDP-encapsulated ESP packet received from the initiator,

UDP destination port : the port 54500.

3.2.4 NAT Keep-Alives

As already specified in Section 3.1.2.4, when HIP or ESP traffics are idle for a long time, keep-alive packets must be sent to prevent the NAT bindings corresponding to the HIP communication UDP channels from timing out and being cancelled.

For a given HIP association, if the UDP encapsulation has been activated, keep-alive mechanisms for both HIP and ESP channels must also be activated. The keep-alive intervals can be set up separately for the two channels. The interval default value is however the same for both types of traffic and is defined to 20 seconds.

For the HIP control UDP channel, basic UDP keep-alive packets as described in Section 3.1.2.4 and Figure 3.4 are sufficient to prevent NAT bindings from timing out. However, keep-alive packets for the HIP channel may also be used for another purpose and require to be related to the corresponding HIP association. In such case the keep-alive packet format would be replaced by HIP UPDATE packets without parameters and UDP-encapsulated as defined in the HIP association. The present NAT traversal extension for HIP does not require specific keep-alive packets for the HIP channel. Consequently HIP keep-alive packets are defined as the basic UDP packets carrying a single-octet payload. The HIP nodes which need to send a keep-alive packet, must use the IP addresses and UDP port numbers specified in the corresponding HIP association to generate the packet.

For the ESP data channel, the generic UDP keep-alive packet for ESP can also be used to keep the NAT bindings alive. The present extension uses however keep-alive packets for another purpose which requires the identification of the corresponding HIP association (see Secion 3.2.5). Thus the specific UDP-encapsulated ESP packet described in Section 3.1.2.4 and Figure 3.5 is used. Further development of the present extension may want to separate real NAT keep-alives from the specific keep-alive packets. In this case, both types of packets would have to be taken into account in HIP specifications and implemented in HIP-aware systems.

When a HIP node has not sent or received secured data from one of its peer for a time exceeding the ESP channel keep-alive interval, it must send a keep-alive packet. To generate this packet, the HIP node must use the outbound SA of the corresponding HIP association.

3.2.5 Mobility Management

During HIP communications, HIP nodes are allowed to modify their location. If a HIP host changes its network connection, it triggers then the update procedure described in Section 2.1.5. In this section, we consider only the case where the mobile host is the host originally located behind a NAT.

When the host behind a NAT changes its location, it has to detect the presence of NATs along the new paths to its peers (using some external mechanism like STUN) before sending any HIP UPDATE message. Alternatively, it may use some heuristics to conclude that it is still located behind a NAT, rather than incur the latency of running the external NAT detection first.

The mobile host must then adapt its behavior to the results of the NAT detection. The two following sub-sections describe respectively the case where the host moves to the public Internet, and the case where the host stays in the same private domain or moves to another private domain but remains behind a NAT.

3.2.5.1 Host Moving to the Public Internet

If the host moves to the public Internet and consequently does not detect any NAT along the new path to one of its peer, the NAT traversal measures are not necessary anymore. Thus the mobile host can send unencapsulated HIP and ESP traffic for the association with that peer.

The mobile host then sends a regular, unencapsulated HIP UPDATE packet to its peer to announce its new location. The peer which detects that the UPDATE packet is not UDP-encapsulated, deduces that NAT-traversal measures have become unnecessary and continues the update procedure with an unencapsulated UPDATE message.

Since UDP encapsulation is disabled, the two hosts must define regular transport-mode ESP SAs during the update procedure. Afterwards, both HIP and ESP traffics are thus transmitted in the regular way, without any NAT-traversal mechanism.

3.2.5.2 Host Moving Behind a NAT

If the mobile host stays on a private network protected by a NAT (either the same network or a new one), NAT-traversal measures are still required. The mobile host triggers the update procedure to announce its new location to its peers. Since NATs are still present, the mobile host must send UDP-encapsulated HIP UPDATE packets, using the port numbers defined in the corresponding HIP associations.

The UPDATE packets will activate the new bindings in the encountered NATs for the HIP traffic. A peer, which receives a UDP-encapsulated UPDATE message, must continue the update procedure with UDP-encapsulation. Since the new location of the mobile host may introduce new bindings in NATs, the peer must use the source IP address and port number of the received UPDATE packet for further HIP exchanges. Moreover, the peer must ignore the possible LOCATOR parameters included in the packet, because they will contain the private addresses of the mobile host and not the proper address for the new UDP channel.

During the update procedure, the mobile host and its peer must define new BEET-mode ESP SAs with UDP-encapsulation. In the same way as the establishment of a HIP communication, the mobile host located behind the NAT can directly define the BEET-mode SAs, but its peer require beforehand a UDP-encapsulated ESP packet to learn the IP address and UDP port number used by the NAT. Therefore, the mobile host must send an ESP channel keep-alive immediately after the end of the update procedure and the definition of the new SAs. The purpose of the keep-alive packet is to generate a new binding in the NAT mapping table, in order to allow the peer to send secured data to the mobile host, but also to indicate to the peer the IP address and port number it must use to send such data.

While the activation of a NAT binding only requires a basic UDP keep-alive packet (Figure 3.4), such a basic packet is not sufficient for the peer to complete its new BEET-mode SAs. The peer must indeed be able to determine from the information contained in the packet, to which HIP association this packet is related, and thus which data channel it reactivates. This purpose is achieved by the specific keep-alive packet represented in Figure 3.5. This issue was the main reason for choosing more complex keep-alive packets for ESP data channels.

Thus the mobile host must send immediately after the update procedure, a single-octet payload in the new ESP data channel. The resulting packet is based on the new outbound BEET-mode SA of the mobile host and encapsulated in UDP. The peer which receives the packet, determines to which HIP association it is related with the SPI value contained in the ESP header of the packet. The SPI value has indeed been defined by the peer itself during the update procedure, and is associated to a specific HIP association. The peer completes then its new BEET-mode SAs with the source IP address and port number of the packet.

3.2.5.3 Other Compatible Scenarios

The update procedures described in the previous sub-sections assumed that the mobile host was the host originally located behind a NAT. These procedures are however also effective in other scenarios.

The first scenario assumes that both HIP nodes are originally in the public Internet. Thus their HIP association is a regular association based on transport-mode ESP SAs and wihtout UDP-encapsulation. If one of the host changes its location and remains in the public Internet, the regular update procedure is performed. However, if it moves from the public Internet to a private network protected by a NAT, the NAT detection results indicate to this host to apply NAT-traversal measures. In this case, the procedure described in the previous sub-section is performed. The UDP-encapsulation is activated, and the peer which receives a UDP-encapsulated UPDATE message concludes that it must also apply NAT-traversal measures. Then the new BEET-mode SAs replace the former transport-mode SAs. The determination of the UDP port numbers to use for the HIP and ESP channels is performed in the same way as the establishment of a new HIP association between an initiator behind a NAT and a responder in the public Internet (procedure described in the previous sections).

In that way, the present specifications allow a mobile host to move several times, successively to the public Internet and to a private network protected by a NAT. Moreover, they allow each host of a HIP association to move from and to any type of network (private or public) under the single condition that their peer is not located behind a NAT at the moment of their location change.

A second scenario where the previous procedures can also be applied, concerns the mobility of the host located in the public Internet. It is indeed possible under specific circumstances, for this host to successfully change its location while its peer is located behind a NAT. The main issue that prevents this scenario from being systematically valid, is the possible presence of restricted cone or symmetric NATs on the path between the HIP nodes. Such NATs would indeed discard the packets coming from the new location of the host, since the new IP address would not be registered in the NAT bindings.

In the case where only full cone NATs are present on the path and the host on the public Internet changes its location but remains in the public Internet, the previous procedures can be applied. The mobile host is still in the public Internet, but since the NAT-traversal measures were required by its peer located behind NATs, the mobile host continues to apply these measures to communicate with its peer. Thus the mobile host sends a UDP-encapsulated UPDATE message to the peer, using the same destination IP address and port number defined in the HIP association. This packet will be properly transmitted to the peer behind NATs, because full cone NATs forward even packets arriving from unknown source IP addresses. Then the update procedure will be completed successfully. The bindings in the NATs will not be modified because the peer behind NATs has not changed its location or the UDP ports it uses. And eventually the HIP communication will continue normally with the NAT-traversal measures activated.

3.2.6 Firewall Traversal

When the initiator or the responder of a HIP association is protected by a firewall, additional issues arise, that may prevent the present NAT-traversal extension from working properly.

When the initiator is located behind a firewall, the NAT traversal mechanisms described in the previous sections depend on the ability to initiate communication via UDP to destination ports 50500 and 54500 from arbitrary source ports, and to receive UDP response traffic from those ports to the chosen source ports.

Most firewall implementations support *UDP connection tracking*. This means that after a host behind a firewall has initiated a UDP communication to the public Internet, the firewall relays UDP response traffic in the return direction. If no such return traffic arrives for a specific period of time, the firewall stops relaying the given IP address and port pair. The NAT-traversal measures defined in the present extension, already enable traversal of such firewalls, if the keep-alive intervals used are less than the refresh interval of the firewall.

If the initiator is behind a firewall that does not support UDP connection tracking, the NAT-traversal mechanisms can still be supported, if the firewall allows permanently inbound UDP traffic from ports 50500 and 54500 and destined to arbitrary source IP addresses and UDP ports.

When the responder of the HIP association is located behind a firewall, the NAT-traversal mechanisms previously described depend on the ability to receive UDP traffic on ports 50500 and 54500 from arbitrary source IP addresses and ports.

Consequently the NAT-traversal measures that have been specified in the present extension, require that firewalls (with or without UDP connection tracking support) allow inbound UDP traffic to ports 50500 and 54500 and allow outbound UDP traffic to arbitrary UDP ports.

3.3 Second Case : Responder Behind a NAT

The previous section has only described the NAT traversal mechanisms for HIP communications with a responder located on a public network. The present section considers the case where the responder may also be located on a private network protected by a NAT.

The main issue of this configuration is that an initiator will not be able to trigger a HIP communication with the responder, because the NAT protecting the responder will discard the incoming messages from the initiator. The solution to solve this problem consists in adding a new middlebox on the public Internet, which will act as intermediary between the HIP node located behind a NAT and other HIP nodes. For this purpose, the new middlebox requires a permanent connection with the host located behind a NAT, to be able to reach it at any moment. Then a HIP node which wants to initiate a communication with the protected host, sends its request to the new middlebox which will relay the request to the protected host, using its specific connection to it.

A similar middlebox has already been introduced in HIP specifications to facilitate the discovery of HIP nodes, and in particular HIP nodes with high mobility. This middlebox is called *Rendezvous Server* and is defined in [18]. The concept of this middlebox is explained in Section 3.3.1 and the enhanced version of this middlebox used for the NAT traversal extension is described in Section 3.3.2.

3.3.1 Rendezvous Server Concept

The purpose of the Rendezvous Server (RVS) described in [18] is to improve reachability and operations when HIP nodes are mobile or multi-homed. Basically, a HIP node willing to use this service, registers to the RVS and indicates which IP address it is currently using. The HIP node keeps the RVS up-to-date about its current location. Furthermore, the DNS records related to the HIP node point to the corresponding RVS. In that way, a host which wants to contact the mobile HIP node will address its request to the RVS which will properly relay the message to the mobile node.

The two following sub-sections describes more precisely the registration procedure, and the role of the RVS in the establishment of new HIP associations.

3.3.1.1 Registration Procedure

The registration to the RVS follows the HIP registration procedure defined in [19]. This procedure is based on the use of an existing HIP association between the registrating host and the RVS, or, if no association exists beforehand, on the establishment of a new one.

If no HIP association exists, the host which wants to register to the RVS service, triggers a common HIP base exchange but includes a registration request in the I2 packet. This request corresponds to a new HIP parameter called REG_REQ. This new parameter indicates to which service the HIP node wants to register, in the present case to the RVS service. The RVS concludes the HIP base exchange with a R2 packet which contains the registration response. The new HIP parameter REG_RESP indicates to the HIP node whether its registration has been validated or denied.

If a HIP association has already been activated between the mobile host and the RVS, the registration request and response are transmitted in HIP UPDATE packets related to the corresponding association.

The two cases of the registration procedure are described in Figure 3.7.

After a successful registration to the RVS, a mobile node modifies the DNS entries related to itself. The node follows the procedures described in [16]. Basically, the node replaces the DNS record pointing to its own current IP address by a record pointing to its RVS. In that way, a HIP node willing to communicate with the registered host, will use the public IP address of the RVS indicated by DNS servers, to contact its peer.

Moreover, when the registered host changes its location, it does not need to update the DNS entries. It just has to keep the RVS up-to-date, using the commmon HIP update procedure with its HIP association with the RVS. The RVS can thus contact the mobile node at its proper location at any moment.

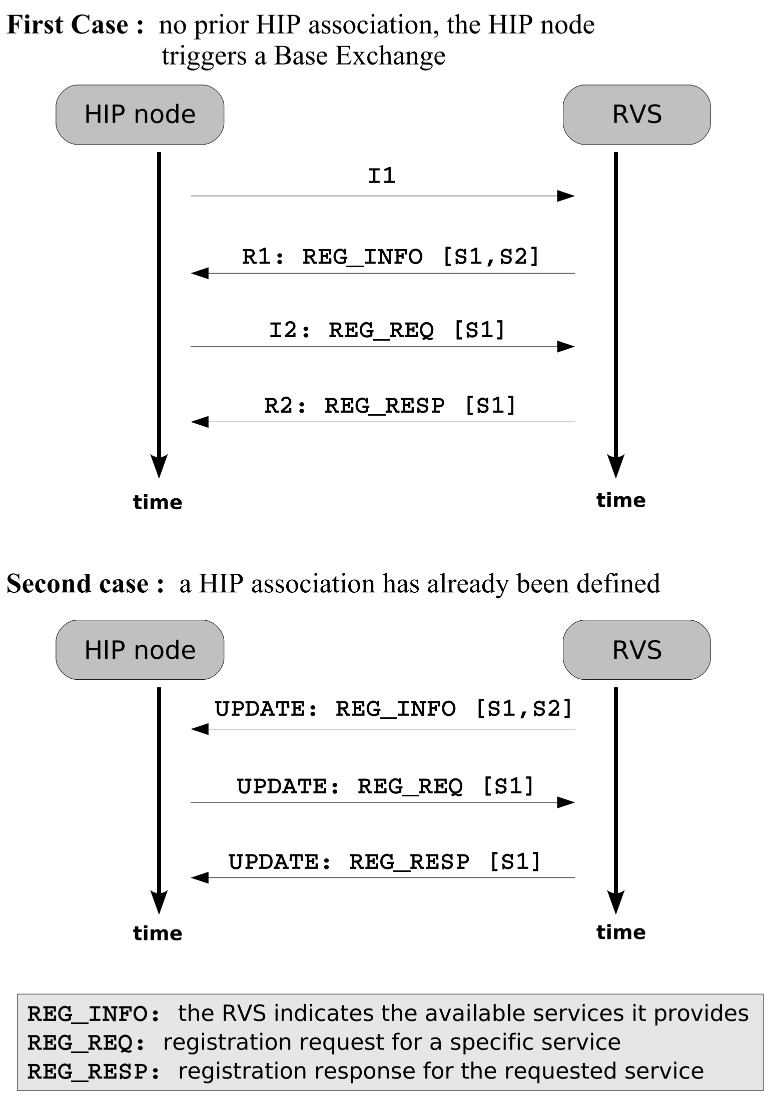
First Case : no prior HIP association, the HIP node triggers a Base Exchange
HIP node
RVS
I1
R1: REG_INFO [S1,S2]
I2: REG_REQ [S1]
R2: REG_RESP [S1]
time
time
Second case : a HIP association has already been defined
HIP node
RVS
UPDATE: REG_INFO [S1,S2]
UPDATE: REG_REQ [S1]
UPDATE: REG_RESP [S1]
time
time
REG_INFO: the RVS indicates the available services it provides
REG_REQ: registration request for a specific service
REG_RESP: registration response for the requested service

Figure 3.7: HIP Registration procedure, included in the HIP Base Exchange or based on UPDATE packets.

3.3.1.2 Establishment of a New HIP Association Relayed by the RVS

When a HIP node wants to initiate a communication with a host registered to a RVS, it triggers the HIP base exchange by sending an I1 packet to the IP address of the RVS. The RVS detects in the I1 packet a destination HIT which does not correspond to one of its own HITs but to one of its registered clients. The RVS then modifies the IP header to relay the I1 packet to the corresponding host. The RVS also includes in the packet a FROM parameter which indicates the original source IP address of the packet. This IP address is then used by the registered host to send packets directly to the host which contacted it. In particular, the remaining packets of the base exchange, R1, I2 and R2, are directly transmitted between the initiator and the registered host.

Consequently the RVS defined in [18] is only used to establish contact between two HIP nodes. The RVS relays only the I1 packet of a base exchange. All other HIP packets and data packets are directly exchanged by the HIP nodes, without going through the RVS.

The HIP base exchange modified by the use of a RVS, is described in Figure 3.8.

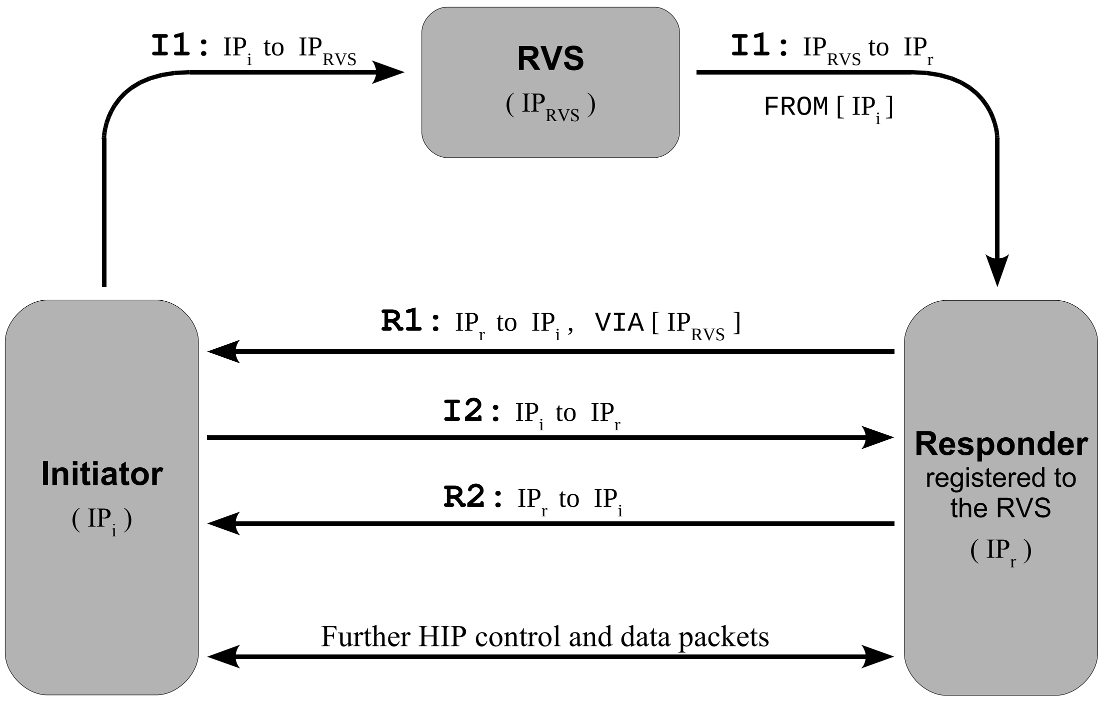

Figure 3.8: HIP Base Exchange, with responder registered to a RVS.

3.3.2 Rendezvous Server for NAT Traversal

A similar concept to the Rendezvous Server described above, will be used to enable the reception of HIP communications when a host is located behind a NAT. The enhanced RVS provides a complete *relay service*. The RVS will indeed be used to relay all HIP control and data traffic. To achieve this mission, the RVS needs to know permanently the location of each HIP node which uses the RVS as relay. For this purpose, the RVS uses the HIP associations already established with the registered hosts and must use a mechanism to record and keep the location of the other HIP nodes up-to-date.

The registration procedure with the RVS and the mechanism to record HIP nodes locations are described in Section 3.3.2.1. The relaying procedures for the RVS are described in Section 3.3.2.2.

3.3.2.1 Registration Procedure and Location Records

The registration to the *relay service* of the enhanced RVS is processed in the same way as the registration to the common RVS. A host can use an already existing HIP association to the RVS and send HIP UPDATE packets to register to the RVS. Or more likely, the host creates a new HIP association by triggering a base exchange. To register to the specific service, the host includes registration request parameters in the UPDATE or I2 packets. The registration validation or denial is then sent by the RVS in an UPDATE or R2 packet, respectively.

In the case where the host is located behind a NAT and does not have any HIP association with the RVS, it must trigger a base exchange with the NAT-traversal measures described in Section 3.2. Since the RVS is located in the public Internet, the NAT traversal mechanism for a host located behind a NAT and responder located in a public network can be applied. The NAT-traversal mechanisms are independent from the parameters present in HIP packets. Therefore, the registration consists in that case in the establishment of a HIP association based on ESP BEET-mode SAs with UDP-encpasulation. The registration parameters are included in the UDP-encapsulated I2 and R2 packets as they would be in non-encapuslated packets.

Moreover, in the case of a registered host behind a NAT, the UDP channels generated by the HIP association will be the channels used by the RVS to relay packets coming from other HIP nodes. Consequently both UDP channels, for HIP control and ESP data flows, must remain active. Thus keep-alive mechanisms must be activated and the corresponding HIP association must not be cancelled.

Since the enhanced RVS will relay all HIP control and data packets, it must also know the location of the HIP nodes in communication with the registered hosts. Thus, if one of the end-hosts of a communication relayed by the RVS has not registered to the RVS service, the RVS has to create a record containing the current location of this node. The RVS must then keep this record up-to-date during the lifetime of the corresponding HIP communication.

Consequently if an unregistered host tries to initiate a communication with a registered host through the RVS, the RVS first checks if it has already an existing record of the location of the initiator, based on its HIT. If a record exists, the RVS updates it if necessary. If no record is found, the RVS establishes a new record, identified by the HIT of the initiator. The RVS stores the source IP address and the possible UDP source port number of the received I1 packet. It can also deduce from the UDP encapsulation or non-encapsulation of the packet, whether or not the initiator is located behind a NAT. After the creation or update of the initiator record, the RVS forwards the I1 packet to the registered host. Moreover, if the initiator changes its location during the HIP communication, the RVS must use the information contained in the UPDATE packets sent by the initiator to correct the corresponding location record.

Then the RVS uses on the one hand the HIP associations established with the registered hosts, and on the other hand the location records created for unregistered nodes, to relay properly the packets of HIP communications between these nodes.

3.3.2.2 RVS Relaying Procedures

The role of the RVS is to relay both HIP control and ESP data traffic. The procedures to relay these two kinds of traffic differ from each other.

To relay a HIP packet properly, the RVS just needs to look at the destination HIT included in the HIP header of the packet. It then looks either for a HIP association or for a location record corresponding to this destination HIT. Three cases are then possible, depending on the presence or absence of NATs between the sender or recipient of the packet and the RVS.

If the incoming packet was not UDP-encapsulated and the final recipient does not require UDP-encapsulation, the RVS must proceed as follows :

- the RVS determines the final recipient of the packet, based on the destination HIT contained in the HIP header,
- the RVS recomputes the HIP header checksum, using the IP addresses stored in the HIP association (for registered hosts) or in the location record (for unregistered hosts) related to the recipient,

- the RVS replaces the current IP header of the packet with a new IP header containing its own IP address as source IP address. The destination IP address is defined as the IP address of the recipient, indicated either in a HIP association or in a location record.

Note that this first case is not likely to happen, unless the registered host moves to a public network and is contacted by an initiator also located in a public network. However, in a such context the RVS would not be required.

If the incoming packet was not UDP-encapsulated but the final recipient requires UDP-encapsulation, the RVS must proceed as follows :

- the RVS determines the final recipient of the packet, based on the destination HIT contained in the HIP header,
- the RVS sets up the HIP header checksum to zero,
- the RVS inserts a UDP header between the former IP header and the HIP header. The destination port is set up to the port used by the recipient and indicated in the corresponding HIP association or location record. The source port is the port used by the RVS, thus the default port 50500. The UDP checksum is computed with the IP addresses of the RVS and the recipient,
- the RVS replaces the IP header of the packet with a new IP header containing its own IP address as source IP address and the IP address of the recipient as destination IP address.

If the incoming packet was UDP-encapsulated, the RVS must keep the UDP-encapsulation, even if it is not required by the final recipient of the packet. The RVS must consequently proceed as follows :

- the RVS determines the final recipient of the packet, based on the destination HIT contained in the HIP header,
- the RVS possibly checks that the HIP header checksum is null, and if it is not the case, sets it to zero,
- the RVS modifies the existing UDP header. If the connection with the recipient requires UDP-encapsulation, the RVS must use the UDP port numbers specified either in the HIP association or in the location record related to the recipient. Otherwise the RVS must use the default port number 50500 for both source and destination ports. It must then recompute the UDP checksum based on its own IP address and the IP address of the recipient,

- the RVS replaces the IP header of the packet with a new IP header containing its own IP address as source IP address and the IP address of the recipient as destination IP address.

After the modifications of the HIP packet, the RVS can forward it to its final recipient. No FROM parameter needs to be included in the HIP packets, as with the common RVS, because every further packet will be relayed by the RVS.

The relaying procedure for ESP packets is slightly different. The ESP packets do not contain the HITs of the HIP communication they are related to. The RVS must then determine the final recipient of the packet, based possibly on the source IP address of the packet, but mainly on the SPI contained in the ESP header. The RVS must consequently be able to establish a relation between a SPI value and a HIP communication it relays. The mechanism to establish this relation is described in Section 3.3.4.

When the RVS knows to which relayed HIP communication the ESP packet is related, it must determine the HIP association or location record corresponding to the recipient of the packet. The RVS must then modify the packet accordingly before forwarding it. The procedure depends on the UDP-encapsulation status of the incoming packet and on the connection with the recipient.

If the incoming ESP packet was not UDP-encapsulated and the final recipient does not require UDP-encapsulation, the RVS just needs to modify the IP header of the packet. It must replace the former IP addresses with its own IP address as source IP address and the IP address of the recipient (indicated in the corresponding HIP association or location record) as destination IP address.

If the incoming ESP packet was not UDP-encapsulated and the connection with the final recipient requires UDP-encapsulation, the RVS must proceed as follows :

- the RVS must insert a UDP header between the former IP header and the ESP header. It must use the UDP port numbers indicated either in the SAs of its own HIP association with the recipient, or in the information recorded about the recipient. The RVS calculates the checksum with the IP addresses also indicated in the SAs or in the recipient record,

- the RVS replaces the IP header of the packet with a new IP header containing its own IP address as source IP address and the IP address of the recipient as destination IP address.

However, the two first cases described above should not happen. In the first case, both end-hosts of the communication do not require UDP-encapsulation, which means that they are both located on a public network. Consequently they do not require the RVS relay service to establish their HIP communication. This first case may however happen if the responder registered to the RVS moves from a private network to the public Internet. In the second case, the recipient of the packet requires UDP-encapsulation. Thus the HIP association between the two end-hosts should have been defined with NAT-traversal mechanisms (see Section 3.3.4). And consequently the packet received by the RVS should already be UDP-encapsulated.

Therefore, the third case correspond to the general scenario for the relay of ESP packets. The incoming ESP packet is already UDP-encapsulated. In that case the RVS must keep the UDP-encapsulation and proceed as follows :

- the RVS modifies the existing UDP header. If the connection between the RVS and the recipient requires UDP-encapsulation, the RVS must use the UDP port numbers specified in the SAs of the HIP association or stored in the record corresponding to the recipient. Otherwise the RVS must use the default port number 54500 for both source and destination ports. It must then recompute the UDP checksum based on its own IP address and the recipient IP address,
- the RVS replaces the IP header of the packet with a new IP header containing its own IP address as source IP address and the IP address of the recipient as destination IP address.

After the headers modifications, the RVS can forward the resulting ESP packet to its recipient.

3.3.3 RVS Databases

The previous section indicated that several types of information are required by the RVS to relay a HIP communication properly. This information must be stored in two databases that the RVS must keep up-to-date.

The first database is the *Clients Database* which contains all the HIP nodes communicating through the RVS. The registered hosts are permanently recorded in this database. The unregistered HIP nodes which communicate with registered hosts, are also listed in the database. Their records are temporary and stored in the database only during the lifetime of their communications.

The records of the clients database contain the following information :

- the HIT of the HIP node, which is used to identify the record,
- the type of the node, either registered or temporary,
- whether or not the HIP node is located on a private network protected by a NAT, thus whether or not NAT-traversal measures are required,

- a pointer to the corresponding HIP association if the node is a registered host,
- otherwise, the IP addresses and UDP port numbers used by the unregistered node for HIP and ESP traffics. The IP addresses are generally the same for both type of traffic but may differ according to the NATs encountered on the path between the node and the RVS. Furthermore, no specific UDP port may be defined if the node is not located behind a NAT. In such case, the common port numbers for HIP and ESP traffic (50500 and 54500) are used when necessary.

The records of the clients database are generated during the registration procedure, for the registered nodes, or during the establishment of a relayed HIP communication for unregistered hosts.

The second database is the *HIP Communications Database*. This database contains all the ongoing communications between two hosts recorded in the clients database. For each HIP communication relayed by the RVS, a record is created and contains :

- the two end-hosts of the communication, identified by their HITs,
- whether or not the communication is UDP-encapsulated,
- the SPI values of ESP flows for both directions,
- a binding between each SPI and the HIT of the corresponding recipient.

The records of this database are created during the establishment of a HIP communication relayed by the RVS. Furthermore, when a HIP association is modified by the mobility of one of the end-hosts, the communications database must be updated accordingly.

The following sections describes more precisely how the RVS database records must be created and updated.

3.3.4 Establishment of a HIP Communication Relayed by the RVS

With help of the two databases described in the previous section, a RVS is able to relay a HIP communication between two HIP nodes. The establishment of their communication is performed as described in the following procedure.

First of all, a host located behind a NAT which wants to receive HIP communications, must register to a RVS relay service. It must then bind its own HIT with the IP address of the RVS in the DNS server records, so that a HIP node willing to communicate with it, sends the request to the RVS.

When a HIP node wants to communicate with a host located behind a NAT, it initiates the communication by sending a I1 packet to the IP address provided by DNS servers, thus to the RVS. The RVS which receives the packet must check that both end-hosts of the communication, identified by the HITs, are recorded in its clients database. If the responder of the HIP communication is not a registered host, then the RVS can not relay the I1 packet and it discards the packet. If the initiator of the communication is not present in the clients database, the RVS must create a new record for this host, based on its HIT. The RVS can already record the IP address and possible port number used for HIP control traffic, and whether or not the node requires UDP encapsulation.

The RVS forwards then the I1 packet to the correct recipient as described in Section 3.3.2.2. The RVS may transmit a UDP-encapsulated I1 packet, even if the packet was originally not encapsulated.

The responder of the base exchange receives the relayed I1 packet and processes it as defined in HIP specifications. It the packet is UDP-encapsulated, the responder must use the NAT-traversal mechanisms described in Section 3.2. It then responds with a R1 packet, sent back to the RVS.

The RVS relays the R1 packet to the initiator of the base exchange. In the case where the responder is located behind a NAT and requires UDP-encapsulation, the initiator receives a UDP-encapsulated R1, although it may have previously sent a non-encapsulated I1 packet. In such case, the initiator must apply the NAT-traversal measures described in Section 3.2 for all further HIP control and data packets.

The HIP base exchange continues with I2 and R2 packets, relayed by the RVS in the same way as I1 and R1. When the RVS receives these new packets, it must use the SPI values included in the HIP parameters to create a record in its communications database. The RVS can read these SPI values because HIP packets are not encrypted. The SPI value indicated in the I2 packet is destined to the ESP traffic from the responder to the initiator, and thus must be bound to the HIT of the initiator in the database record. Similarly, the SPI value indicated in the R2 packet is destined to the ESP traffic from the initiator to the responder and must be bound to the responder's HIT. After the transmission of the R2 packet, the communications database record is completed and validated.

After the end of the base exchange and the establishment of a HIP association between the end-hosts, the initiator sends a first ESP packet. The RVS receives the packet and deduce from the SPI contained in the ESP header to which HIP association it is related. If necessary, the RVS use this packet to complete the clients database record corresponding to the initiator, with the IP address and port number used for the ESP traffic. It forwards then the packet to the responder as described in Section 3.3.2.2.

Then, the RVS relays further HIP and ESP packets as described in Section 3.3.2.2. If one of the end-hosts of the HIP association is located behind a NAT, the NAT-traversal measures have been enabled during the base exchange, and the end-hosts must continue to follow the specifications of Section 3.2. Otherwise the end-hosts follow the common HIP specifications for the continuation of their communication.

3.3.5 Mobility Management

The previous sections have described how a HIP communication can be established with a host located behind a NAT, and relayed by a RVS. Some modifications in the communication parameters may however be required if one of the end-hosts changes its location, and in particular in the configuration of the RVS. The RVS needs indeed to update its databases according to the new location of the mobile host. Two cases must be differentiated depending on the moving host and whether it is registered or not to the RVS relay service.

3.3.5.1 Mobility of an Unregistered Host

If an unregistered initiator of a HIP communication relayed by a RVS changes its location, it uses the HIP update procedure to announce its new location to its peer. If the initiator moves from or to a private network protected by a NAT, or if the responder is located behind a NAT, the initiator must use the specifications described in Section 3.2.5. Otherwise the initiator uses the common HIP specifications.

In both cases, the initiator sends an UPDATE packet to the responder's RVS which will relay the packet to the responder. When the RVS receives an UPDATE packet from the initiator, it must correct the IP address and UDP port number for HIP traffic in the clients database entry corresponding to the initiator. If the UPDATE packet contains a new SPI value, the RVS must also correct the corresponding communications database entry accordingly.

If the UDPATE packet is not UDP-encapsulated and contains LOCATOR parameters, the RVS must remove these parameters. This prevents the responder from using the initiator IP address instead of the RVS IP address to send back some data. This measure is not necessary when the packet is UDP-encapsulated, since the specifications described in the previous sections indicate to the responder to not take the LOCATOR parameters of encapsulated packets into account.

After this possible modification of the UPDATE packet, the RVS relays it to the responder, using the procedure defined in Section 3.3.2.2. The responder processes it according to usual specifications and sends an UPDATE back to the RVS. The RVS uses the updated record of the initiator to relay the packet properly. If this second UPDATE packet contains a new SPI value, the

RVS must also correct the corresponding communications database entry.

The following UPDATE packets (there may be one or two of them) are then relayed as common HIP packets by the RVS. The possible new SPI values they contain, are stored in the communications database.

Finally, the RVS uses the first ESP packet sent by the initiator to update the clients database and register the correct IP address and UDP port used by the initiator for ESP traffic.

3.3.5.2 Mobility of a Registered Host

A HIP node registered to the RVS may also change its location while having HIP communications with other nodes. This host is generally the responder of HIP communications, but it may also be the initiator in the case where both hosts have registered to the same RVS.

If a registered host changes its location, it must first update its HIP association with the RVS, so that the RVS is able to relay further packets to other HIP nodes properly. After the update of the communication with the RVS, the mobile host can trigger the update procedures for HIP communications relayed by the RVS.

The registered host updates its association with the RVS with the procedures defined in Section 3.2.5, or in common HIP specifications if the host was and stays in a public network.

The registered host then updates the HIP communications relayed by the RVS. For this purpose, it sends UPDATE packets to the RVS which will relay them to the correct peers. As described in the previous sub-section, the RVS removes possible LOCATOR parameters contained in the packets, if necessary. The RVS also updates its communications database with the possible new SPI values. The clients database does not need specific update, because the parameters used to relay packets to the mobile host have been corrected in the corresponding HIP association beforehand.

Thus in the case of a registered mobile host, the RVS only requires an updated HIP association with this host. Then the RVS relays HIP and ESP packets as described in previous sections, and possibly, updates its communications database if new SPI values are provided.

3.3.6 Keep-Alive Mechanisms

The RVS has to be located in the public Internet where it can be reached by any HIP node. Consequently it is not direclty concerned by NAT binding keep-alive mechanisms. This is however not the case for HIP nodes located behind NATs and communicating through the RVS. Therefore, the RVS is likely to receive keep-alive packets from a HIP node, either registered or unregistered. Since the purpose of such packets is to keep the bindings in the NATs present between the hosts and the RVS alive, the RVS does not need to relay them to the other end-host of a relayed communication. Consequently, the RVS can simply discard the incoming keep-alive packets.

3.3.7 Open Issues

The previous sections have described some mechanisms to allow HIP nodes to establish and receive HIP communications when they are located on a private network protected by a NAT. The described specifications are sufficient to allow HIP communications in almost every scenario including NATs between the end-hosts. However, some issues still remain in some particular cases. The first issue concerns the efficiency of the RVS relay service. The second issue is related to the use of IPsec ESP for the secured data transfer of HIP communications. Indeed, the use of SPI values in the RVS may become problematic under particular circumstances.

These two main issues are described in the further sub-sections.

3.3.7.1 RVS Efficiency Issue

The NAT-traversal mechanisms described in the previous sections, indicate that a HIP communication triggered through a RVS (with a responder located behind a NAT), is afterwards completely relayed by the RVS. The enhanced RVS relays both HIP and ESP traffic, and not only the first I1 packet of a HIP base exchange.

This difference with the basic RVS was introduced to solve the NAT traversal problems that could appear if only the I1 packet was relayed by the RVS. Indeed a HIP base exchange between two end-hosts both located behind a NAT, is likely to fail if the RVS relays only the I1 packet to the responder. This scenario is described in Figure 3.9.

In the particular case where both initiator and responder of a HIP communication are located behind a NAT, the following problem can appear. The initiator sends a I1 packet to trigger the base exchange. The packet is sent to the IP address of the responder's RVS, but it traverses beforehand the NAT of the initiator. Thus the NAT creates a new binding with the IP address and UDP port number used by the initiator, its own public IP address and a port number of its choice. If the NAT is a restricted cone NAT (or more restrictive), it also records the IP address of the RVS. The NAT forwards then the packet to the RVS with its own IP address.

The RVS then introduces a FROM parameter in the packet and relays it to the responder. The FROM parameter contains the public IP address of the initiator's NAT. Thus, the responder which receives this packet will reply with a R1 packet that it sends to the IP address provided in the FROM parameter. The R1 packet is processed by the responder's NAT and then forwarded to the

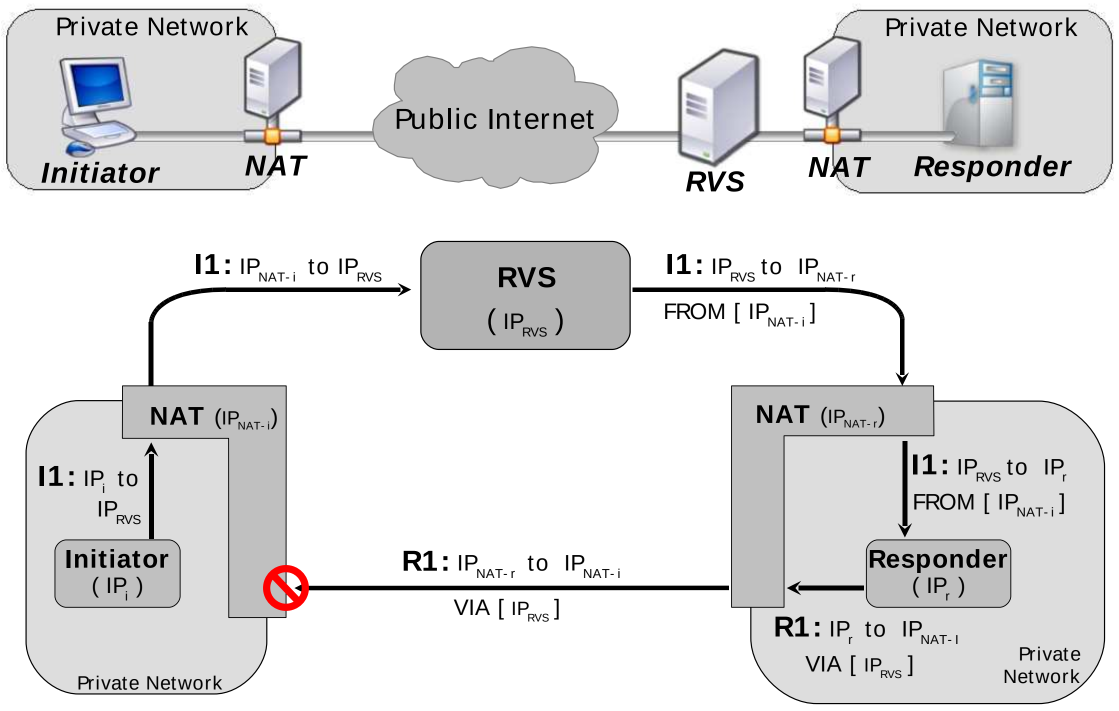

Figure 3.9: Basic RVS issue. Failure of the HIP Base Exchange when both end-hosts are located behind a NAT.

initiator’s NAT.

When the initiator’s NAT receives the R1 packet, the source address of this packet corresponds to the public IP address of the responder’s NAT. If the NAT is a restricted cone NAT, it will discard the incoming R1, because the source address recorded for the NAT binding was the IP address of the RVS and not the public IP address of the responder’s NAT. Consequently the HIP base exchange fails and the HIP association is not established.

For this reason, the RVS relays the whole HIP communication and not only the I1 packet of the base exchange. In that way the possible rejection of HIP packets by the initiator’s NAT are avoided. Relaying the whole HIP communication is thus a NAT-traversal solution which works properly in every configuration.

However, this mechanism may not be the most efficient one in several situations. The general case of unefficiency consists in a HIP communication triggered by an initiator located in the public Internet and the responder’s RVS which is not located on the direct path between the initiator and the responder’s NAT. In a such configuration, after the RVS relayed the I1 packet, the responder could directly send the R1 packet to the initiator. This packet would create the necessary binding for the HIP traffic in the responder’s NAT. Thus the base exchange could be completed without the relay of the RVS. The responder would then have to send a ESP packet directly after the base exchange, to create the corresponding binding in its NAT. The following HIP communication could then be handled in the same way as the case described in Section 3.2, without the help of the RVS. Consequently if the RVS is not on the direct path between the initiator and the responder, relaying the whole communication through the RVS introduces an unnecessary complexity.

The current specifications of the NAT traversal extension consitute consequently a valid solution for the case of a responder located behind a NAT. But the complete relay of HIP communications through the RVS is not the most efficient solution in many cases and should therefore be improved in further development of the NAT traversal extension.

3.3.7.2 SPI Management Issue in RVS

As already mentioned, a second issue may also occur in the RVS, due to the use of IPsec ESP for the secured data exchanges. The specifications of the enhanced RVS indicate that the RVS uses the SPI values of the incoming ESP packets to determine to which HIP association they are related. However, under particular circumstances, the RVS may not be able to identify this HIP association with the single SPI value. An example of problematic situation is described below.

Two different HIP nodes are located in a private network protected by a NAT and have registered to the same RVS. These hosts could also be located in two different private networks, but they must use the same RVS. Two other HIP nodes are located in the public Internet and each one establishes a HIP communication with one of the first hosts. Both communications are relayed by the RVS. The configuration is represented in Figure 3.10.

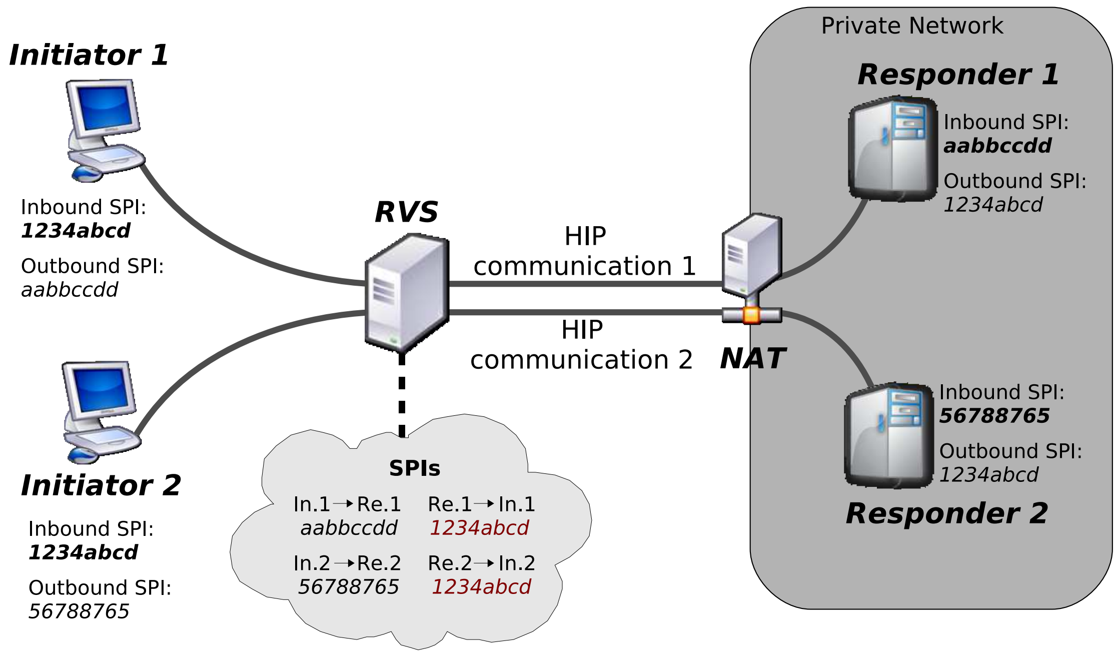

Figure 3.10: SPI management issue in the RVS.

During the HIP base exchange, each host chooses the SPI value of its own inbound security association. Nothing prevents the two hosts located in the public Internet to choose the same SPI value. Consequently, the two ESP traffics relayed by the RVS and going from the private network to the public Internet are related to two different HIP communications but use the same SPI. Thus if one of the host located behind the NAT sends a ESP packet to its peer, the RVS will find two correct HIP associations matching the SPI value contained in the packet. Since the ESP packet does not contain any further information to identify the flow, the RVS will not be able to determine the proper recipient of the packet.

The RVS could possibly use the IP addresses and UDP ports of the incoming ESP packet to determine more precisely the HIP association it is related to. But this may not always solve the problem because nothing prevents the two hosts of the previous scenario from being in fact the same single host. And in a such case, the two ESP traffics emitted by the host behind the NAT would have the same SPI, IP addresses and UDP ports at the RVS.

Although the described scenario is very unlikely, some mechanisms should be defined in further specifications to handle this problem. A possible solution would be to include new procedures allowing the RVS to alert a HIP node that it is about to define a security association with a SPI value already used by another active HIP communication. Thus the concerned HIP node would have to choose another SPI value for the inbound SA it is establishing.

Chapter 4

Implementation

As already mentioned, three main projects are currently ongoing to implement HIP specifications. For the present thesis, we worked on one of these projects, namely the OpenHIP project. This project is mainly developed by the Boeing Company. The main contribution in this project has consisted in the implementation of the NAT traversal extension described in the present thesis.

4.1 OpenHIP Overview

The OpenHIP project is destined to MS Windows and Linux platforms. It is therefore divided in several branches with a main part common to the various implementations. For this thesis, we mainly focused on the OpenHIP implementation for Linux operating systems, which is itself available in two different versions. The first version is a *kernel* implementation (described in Section 4.1.1) and the second, a *user-space* implementation (described in Section 4.1.2). Both versions are developed in C language.

4.1.1 Kernel Implementation

The main purpose of the kernel implementation is to dispose of an efficient implementation for Linux operating systems. This implementation therefore tries to use as much as possible the features of the Linux kernel.

In particular the secured data transfers of HIP communications are based on IPsec ESP and use the IPsec stack of the Linux kernel. To manage the security associations (SAs) and policies (SPs), the kernel implementation uses the *libipsec* library and the *setkey* utility which are components of the *IPsec-Tools* project.

In this implementation, a HIP-daemon running in user-space interacts directly with the network stack of the Linux kernel. This interaction requires however some modifications of the kernel itself. The OpenHIP project provides therefore a specific patch for the Linux kernel.

This implementation is meant to be efficient but has a major disadvantage. With this version of OpenHIP, applications continue to use real IP addresses to contact their peers. They do not use the HITs or LSIs supposed to replace IP addresses in the transport and application layers. The relation between host identities and IP addresses is indeed only present in the HIP daemon which communicates directly with the kernel.

4.1.2 Userspace Implementation

The second version of OpenHIP for Linux operating systems consists in a user-space implementation. This means that the major part of network and cryptographic processing related to HIP is accomplished in user-space. Indeed a HIP service running in user-space includes a HIP daemon which manages the HIP associations, but also a complete IPsec stack separated from the kernel.

On the contrary to the kernel implementation, the applications establishing HIP communications with the user-space implementation of OpenHIP, must use directly the host identifiers, and in particular the LSIs for IPv4-based applications. To achieve this goal, the implementation uses a *TUN/TAP device*, which is a virtual network device, provided by the Linux kernel. Since the LSI domain corresponds to the IP addresss domain 1.*.*.*/8, the TUN/TAP device is configured to handle all data corresponding to this network domain, and the IP address of the virtual device is set up to the LSI of the local host.

The main structure of the user-space implementation is described in Figure 4.1.

Applications establishing HIP communications with other HIP nodes, use LSIs to communicate. The transmitted data corresponds therefore to the virtual domain 1.*.*.*/8 and is handled by the TUN/TAP device in the kernel. The outgoing data is copied from the virtual device to user memory and processed by the HIP service, before being sent back to the real network interface in the kernel. The incoming data follows the inverse path through the kernel, the HIP service, the virtual device, and is then sent back to the corresponding applications.

These multiple exchanges of data between kernel and user-space imply therefore a reduced performance of the implementation. However, the user-space implementation of OpenHIP is much more flexible and modular than the kernel implementation. Since the whole processing related to HIP is done in the HIP service, new extensions or modifications of the implementation are not limited by kernel requirements. Furthermore, this implementation is more adapted to the main concept of HIP, because applications have to use the new host identifiers, while real IP addresses

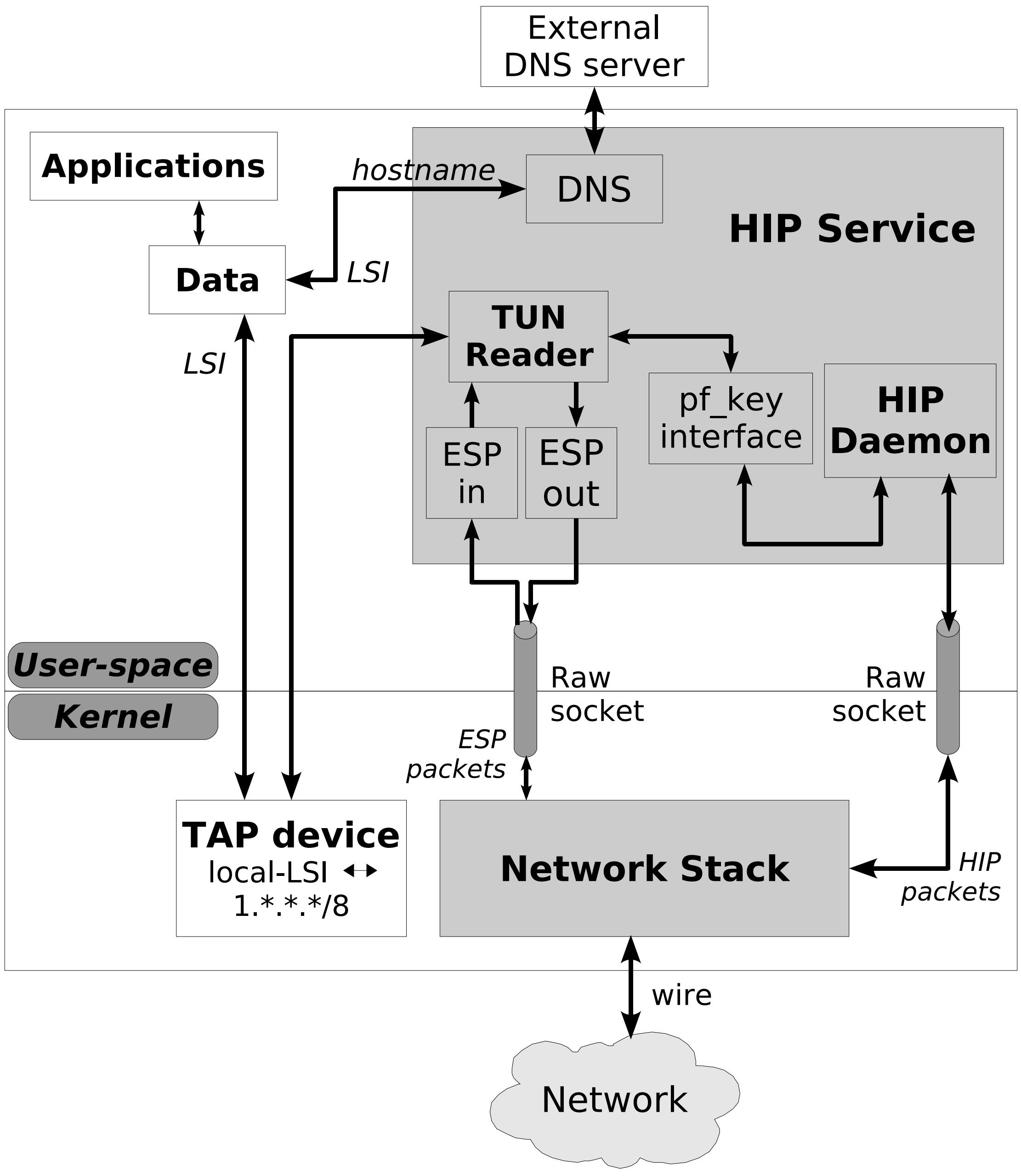

Figure 4.1: Structure of OpenHIP user-space implementation for Linux platforms.

are only manipulated by the HIP service. For these reasons, the user-space implementation was chosen to implement and test the NAT-traversal extension for HIP specified in the present thesis.

4.2 Integration of the NAT Traversal Extension in OpenHIP

The integration of the NAT-traversal support has required modifications in almost every part of the HIP service processing. The main modifications implemented during the thesis are listed below.

First of all, the NAT traversal mechanisms have been implemented in the HIP daemon and the IPsec stack of the HIP service. UDP encapsulation and decapsulation mechanisms have been implemented for the HIP control traffic. A new UDP socket set up to the port number 50500 has been added to send and receive UDP-encapsulated HIP packets.

The HIP daemon is responsible for the establishment of security associations in HIP communications. The use of BEET-mode ESP SAs requires therefore some improvement in the SA management. Since the HIP service has its own IPsec stack, the ESP processing has also been improved to support and handle properly BEET-mode SAs. UDP encapsulation and decapsulation mechanisms for ESP traffic were also required, and a new UDP socket was established on port 54500.

For simplicity and efficiency reasons, the enhanced implementation uses always the UDP port numbers 50500 and 54500 to establish new UDP-encapsulated HIP associations. This avoids the creation and management of multiple UDP sockets for HIP communications.

Moreover, some keep-alive mechanisms have been implemented to assure that the NAT bindings corresponding to the new UDP channels do not time out during the lifetime of HIP communications.

With these first modifications, HIP communications can be established successfully with an initiator located behind a NAT and a responder in a public network. Some improvements are then required to determine the presence or absence of NATs and adapt the use of NAT-traversal procedures accordingly.

A NAT detection mechanism has been integrated to the OpenHIP project, based on the STUN protocol. To determine whether it is located behind a NAT or not, a HIP node needs therefore a STUN client. The open-source project *STUN Client and Server library*, provides a STUN client developed in C++ language. This client has been converted in C language and integrated in the OpenHIP project.

A HIP node can consequently use the result provided by the STUN client and enable or disable NAT-traversal mechanisms depending on the presence or absence of NATs.

Finally, the mobility management, bound to the NAT detection system, has been improved in the implementation, so that a host can move from or to a private network as much as it wants, while its peers remain in a public network. The use of UDP-encapsulation and BEET-mode SAs is then limited to the necessary cases.

All the previoulsy mentioned enhancements fulfil therefore the specifications described in Section 3.2. To be able to establish HIP communications with a responder located behind a NAT, the enhanced RVS described in Section 3.3 is required. The RVS functions have not been integrated in the OpenHIP project yet. Thus the specifications of NAT-traversal mechanisms can be validated with the current OpenHIP implementation, only in the case of an initiator located behind a NAT.

4.3 Evaluation

The NAT-traversal extension implemented in the OpenHIP project, requires some tests to confirm that the HIP service works properly. Successful tests would also validate the specifications and design choices made for the NAT-traversal extension.

The first test consists in verifying the proper use of BEET-mode SAs and UDP-encapsulation of both HIP and ESP traffic, when the initiator of a HIP communication is located behind a NAT. This corresponds to the basic requirements of the NAT-traversal extension. An implementation can therefore be considered as valid only if a such test is successfully passed. This is the case for the OpenHIP implementation with the NAT-traversal extension. A UDP-encapsulated HIP communication can successfully be performed between an initiator behind a NAT and a responder in a public network.

The second test consists in the validation of mobility scenarios. Since no RVS is currently available to relay HIP communications to responders located behind a NAT, the mobility scenarios can only be tested with a mobile host communicating with a responder located in a public network. A specific test is described below and in Figure 4.2.

In this test, the network 10.17.1.*/24 represents a public network. A responder with IP address 10.1.2.53 is directly connected to this public network. The initiator of the HIP communication is located initially in position 1, in the private network 192.168.0.*/24. This private network is connected to the public network through the full-cone NAT *NAT-1*, with public IP address 10.17.1.1. A second private network, 192.168.10.*/24, is also connected to the public network through a full-cone NAT, *NAT-2*, with public IP address 10.17.1.21. In addition, a STUN server is directly connected to the public network and disposes of two IP addresses 10.17.1.3 and 10.17.1.4.

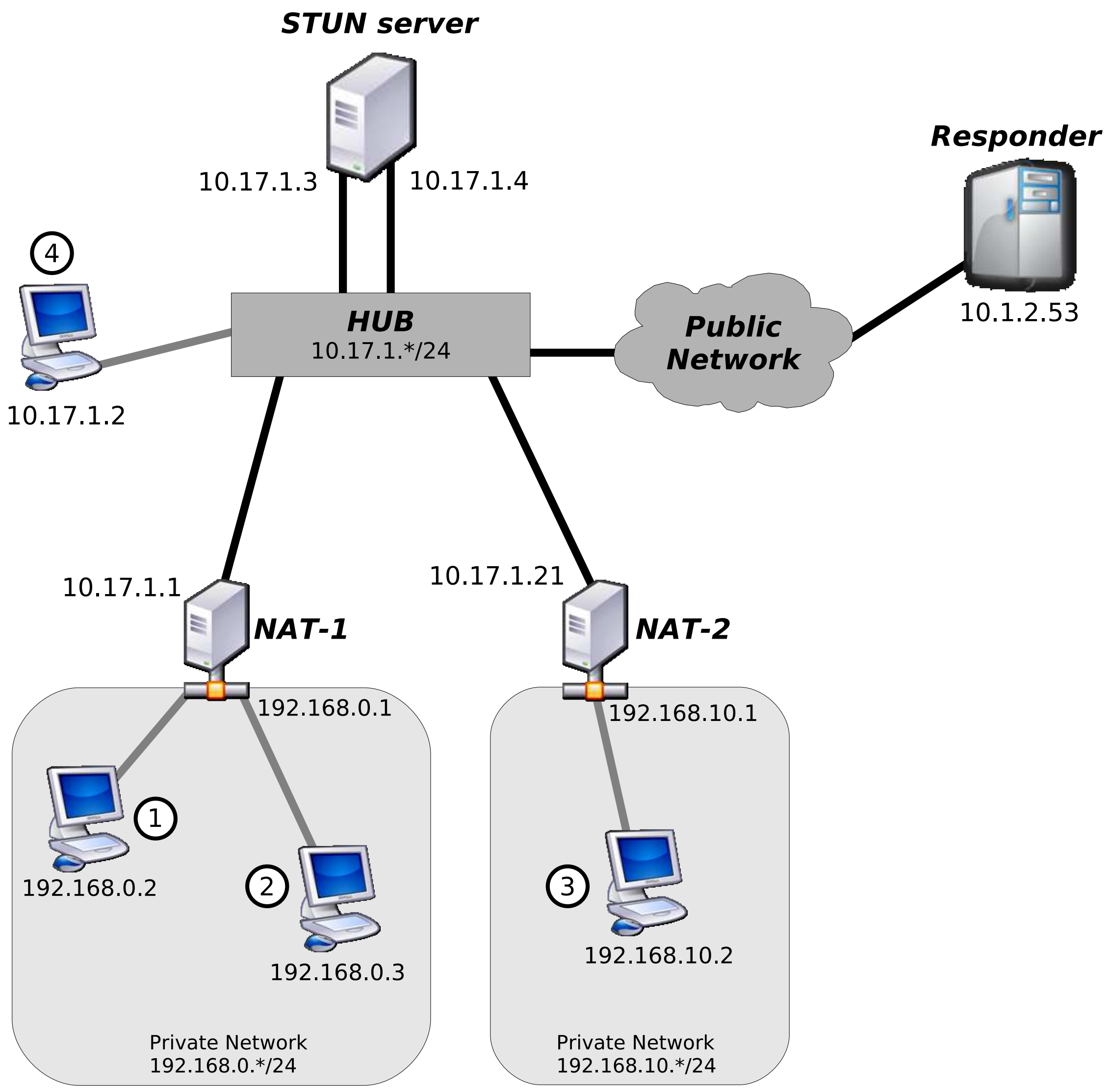

Figure 4.2: Mobility test for the OpenHIP implementation with NAT-traversal extension. Initiator moving to various locations in public or private networks.

The initiator triggers the HIP communication with the responder from location 1, with private IP address 192.168.0.2. After the successful establishment of the communication, the initiator begins the transfer of a big file to the responder, through the HIP communication. The transfer program used is the Secured Shell (SSH) transfer utility *scp*. During the file transfer, the initiator moves to one of the following locations :

- Location 2. The initiator changes its location inside the same private network and gets the new IP address 192.168.0.3. From the responder point of view, only the UDP port number used by the NAT will be modified. The HIP and ESP packets of the HIP communication still have the same source IP address. Furthermore, the NAT-traversal mechanisms remain activated.

- Location 3. The initiator moves to the second private network and uses the IP address 192.168.10.2. It is still located behind a NAT and consequently keeps using NAT-traversal mechanisms. From the responder point of view, both IP address and UDP port used by the initiator's NAT will be modified.

- Location 4. The initiator moves to the public network and gets the public IP address 10.17.1.2. Consequently NAT-traversal mechanims are disabled, and HIP control and data packets are not UDP-encapsulated anymore.

After each location change, the initiator performs a NAT detection test with the STUN server. Based on the results of the test, the initiator then keeps or disables NAT-traversal mechanisms and triggers the HIP update procedure. After the update of the HIP communication, the file transfer, temporarily stopped during the mobility phase, is automatically resumed and continues properly.

In each case, the initiator's mobility is properly handled. Furthermore, after the initiator successfully moved to location 2, 3 or 4, it can come back to its previous location 1 or move to another of the described locations, with the file transfer still running. If necessary, the initiator reactivates NAT-traversal mechanisms. In all these cases, the initiator's mobility is also successfully handled.

Even when several location changes are performed during the file transfer, the update procedures are successfully processed and the file is finally successfully transfered to the responder.

The same tests have been performed with a file transfer triggered by the responder. In these scenarios too, no problem was detected and the file transfer ended successfully.

Consequently the described test procedure, which gave positive results, validates the mobility support of the OpenHIP implementation with NAT-traversal extension.

Further tests with multiple HIP nodes have also been performed. The first scenario consists in two HIP nodes located in the same private network and establishing simultaneously a HIP communication with the same responder located in a public network. The second scenario consists in one HIP node located in a private network and establishing communications with two hosts located in a public network. These two hosts can also establish a HIP communication between each other. In both scenarios, the mobility of the hosts initially located behind a NAT is tried out. Every test has been successfully passed.

All the successful tests performed, prove therefore the good functioning of the NAT-traversal extension integrated in the OpenHIP project. Moreover, they testify to the validity of the specifications described in the present thesis, and in particular in Sections 3.1 and 3.2.

Eventually, some additional tests could still be performed to evaluate the scalability of the OpenHIP implementation in general, and more specifically from the NAT-traversal extension point of view. Such tests would include a multitude of HIP nodes establishing HIP communications between each other, some of them located in a public network, while others would be located behind one or several NATs. The different types of NATs described in Section 2.3.1, should also be represented in these tests.

Chapter 5

Related and Future Work

The HIP protocol is still under development, and some further work is required. In Section 5.1, we describe the open issues that have to be solved, and the future work directly related to the protocol specifications. In Section 5.2, we present further requirements concerning the HIP implementation projects, and in particular the OpenHIP project.

5.1 Protocol Development

The present thesis has provided specifications for enabling NAT-traversal support in HIP communications. The main part of these specifications, described in Sections 3.1 and 3.2, have been submitted to the IETF HIP Working Group and adopted as basis for ongoing standardisation of the NAT-traversal mechanisms for HIP communications.

The NAT-traversal specifications for the case of a HIP communication between an initiator located behind a NAT and a responder in the public Internet, have been submitted in the following IETF Internet-Draft :

HIP Extensions for the Traversal of Network Address Translators
(draft-schmitt-hip-nat-traversal-00, work in progress)
V.Schmitt, A.Pathak, M.Komu, L.Eggert, M.Stiemerling. February 2006.

As mentioned, this document is the basis for standardised specifications. Consequently the present specifications have to be developed, improved and described in further versions of the document. Although the main concept of NAT-traversal mechanisms is durably defined in the present thesis, some design choices may me modified in future versions of the specifications. Among the discussed points, there is already the decision about using a single common UDP channel for both HIP control and data traffic, or two separated UDP channels (as described in the current specifications).

From this choice about UDP channels, depend two further issues. First the choice of the general UDP port numbers for UDP-encapsulated HIP control and data packets (thus either one or two port numbers, and dedicated to HIP or shared with IKE/ESP protocols). The second issue concerns the specification of keep-alive and reactivation mechanisms for the corresponding UDP channels.

Another minor issue concerns the role of LOCATOR parameters used in HIP control packets. These parameters carry indeed wrong or unnecessary information in scenarios assuming the presence of NATs between HIP nodes.

Beside these minor issues, a much more important point to develop in future specifications, is the role of the Rendezvous Server described in Section 3.3. The specifications of the RVS presented in this thesis still need to be approved by the HIP Working Group and possibly integrated to the corresponding IETF documents.

Moreover, as mentioned in Section 3.3, the proposed specifications of the RVS are not the most efficient ones in several cases. Some improvements are therefore required. From a general point of view, the NAT-traversal mechanisms should be enhanced in a way taking the number and type of encountered NATs into account. In particular, full-cone NATs, which are less restrictive than other types of NATs, should be handled separately. Besides, the RVS specifications should be adapted to the type of detected NATs, and the relay of whole HIP communications should be enabled only in necessary cases.

Eventually, the management of complex mobility scenarios should be precisely described in the protocol specifications. An example of such advanced scenario is presented in Figure 5.1.

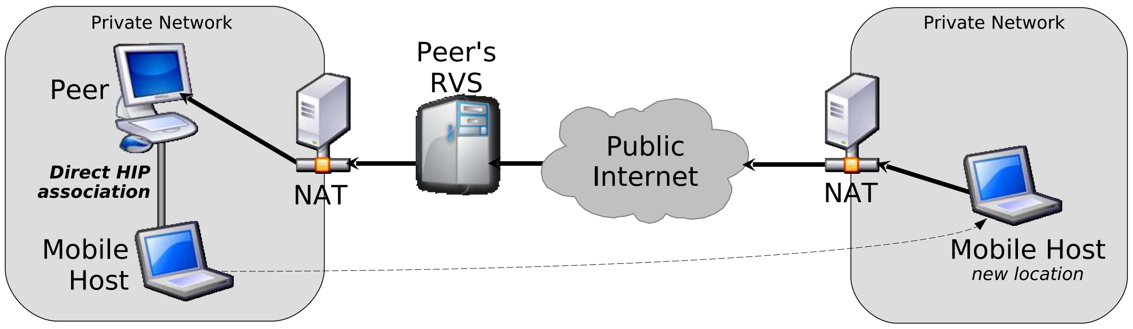

Figure 5.1: Example of a complex mobility scenario for HIP communications.

In this scenario, two HIP nodes located in the same private network, connected to the public Internet through a NAT, establish a HIP communication between each other. Since the two hosts are directly connected, the NAT-traversal mechanisms should be disabled. One of this host is a mobile host and modifies its location by moving to another private network protected by a NAT. In a such case, the HIP specifications should indicate to the mobile host the following procedure : determine the location of its peer's RVS in the public Internet, activate the NAT-traversal mechanisms and contact its peer through the RVS to update their HIP association. Afterwards the HIP communication between the two hosts would traverse their respective NATs and would possibly be relayed by the RVS, according to the types of NATs.

To handle such complex scenarios, specific mechanisms should therefore be developed to determine precisely in which cases NAT-traversal mechanisms are required, when a RVS needs to relay a whole communication, and how a mobile host can reach all its peers after it changed its location.

5.2 Implementation

The previous section has described the future work concerning the specifications of the NAT-traversal extension for HIP communications. Beside these theoretical considerations, the implementations of the HIP protocol also require improvements.

The implementation project considered in this thesis was the OpenHIP project. As mentioned in Chapter 4, the enhanced RVS relaying complete HIP communications, has not been implemented yet. This is therefore the first enhancement of the implementation that should be considered. The RVS is indeed a major improvement in HIP node connectivity.

The previous section also stated that modifications in the current specifications may happen before the final standardisation of the protocol. The OpenHIP project will consequently have to take such modifications into account and be enhanced to keep its conformance to the specifications. In particular the possible modifications of the NAT traversal extension and the introduction of the enhanced RVS in the IETF documents would have to be passed on the implementation.

Finally, as already mentioned, three main projects are under development to implement the HIP protocol. These projects do not currently focus on the same aspects of the protocol. In particular the NAT-traversal extension has not been implemented in the two other projects yet. However, in the long term, the various implementations should provide the same main features, based on the standardised specifications. Thus, eventually, the interoperability of the implementations should be checked. And especially for the case of networks including NATs, the various implementations should perform compatible NAT-traversal mechanisms, based on the extension specified by the HIP Working Group.

Chapter 6

Conclusion

In the present thesis we have presented specifications for a NAT-traversal support in HIP communications.

In Chapter 2, we presented the main protocol specifications of Host Identity Protocol (HIP) and IPsec Encapsulating Security Payload (ESP). We also described the functioning of Network Address Translators and the issues they introduce in HIP communications.

In Chapter 3, we proposed an extension of the protocol enabling NAT-traversal support for HIP communications. This extension was based on the UDP-encapsulation of HIP control and data traffic and on the use of ESP BEET-mode Security Associations. We also introduced a new device, the Rendezvous Server (RVS), to allow the reception of HIP communications in private networks protected by a NAT.

In Chapter 4, we described the work accomplished to integrate the proposed extension into one of the main HIP implementation project, namely OpenHIP.

And finally, in Chapter 5, we discussed some points that should be developed to improve the protocol specifications and the corresponding implementations.

The main contribution of this thesis can therefore be divided in two parts. The major contribution consists in a theoretical approach of the HIP protocol. We proposed specifications for a HIP extension providing NAT-traversal support. The main part of these specifications have been submitted as an Internet-Draft to the IETF in February 2006. This document has been adopted by the IETF HIP Working Group as the starting point of the standardisation of a NAT-traversal extension for HIP. The present thesis also proposed some further specifications for an enhanced Rendezvous Server, improving HIP node reachability.

The second main contribution of this thesis consists in a more practical approach. Although the enhanced RVS has not been implemented yet, the main part of the NAT-traversal extension has been successfully integrated to the OpenHIP project. The successful implementation of the NAT-traversal extension is an important improvement in itself. But it also testifies to the feasibility of the proposed solution and acts consequently as a validation of the specifications adopted by the HIP Working Group.

To conclude, the HIP protocol is still under development and a lot of work needs to be accomplished before its specifications can be recognised as a standard. However, the protocol enhancement proposed in the present thesis, constitutes a good basis for further developments. The NAT-traversal extension will be indeed an important step to integrate HIP communications in nowadays networks. This would be particularly true with the multitude of private networks connected to the public Internet through NATs.

Bibliography

[1] J. Postel. *Internet Protocol*. RFC 791. September 1981.

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

[3] J. Postel. *Transmission Control Protocol*. RFC 793. September 1981.

[4] J. Postel. *User Datagram Protocol*. RFC 768. August 1980.

[5] S. Kent and R. Atkinson. *Security Architecture for the Internet Protocol*. RFC 2401. November 1998.

[6] S. Kent and R. Atkinson. *IP Authentication Header*. RFC 2402. November 1998.

[7] S. Kent and R. Atkinson. *IP Encapsulating Security Payload (ESP)*. RFC 2406. November 1998.

[8] P. Srisuresh and K. Egevang. *Traditional IP Network Address Translator (Traditional NAT)*. RFC 3022. January 2001.

[9] B. Aboba and W. Dixon. *IPsec-Network Address Translation (NAT) Compatibility Requirements*. RFC 3715. March 2004.

[10] J. Rosenberg, J. Weinberger, C. Huitema and R. Mahy. *STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)*. RFC 3489. March 2003.

[11] A. Huttunen, B. Swander, V. Volpe, L. DiBurro and M. Stenberg. *UDP Encapsulation of IPsec ESP Packets*. RFC 3948. January 2005.

[12] C. Kaufman. *Internet Key Exchange (IKEv2) Protocol*. RFC 4306. December 2005.

[13] R. Moskowitz and P. Nikander. *Host Identity Protocol Architecture*. Internet-Draft draft-ietf-hip-arch-03, Work in Progress. August 2005.

[14] R. Moskowitz, P. Nikander, P. Jokela and T. Henderson. *Host Identity Protocol*. Internet-Draft draft-ietf-hip-base-05, Work in Progress. March 2006.

[15] P. Jokela, R. Moskowitz and P. Nikander. *Using ESP transport format with HIP*. Internet-Draft draft-ietf-hip-esp-02, Work in Progress. March 2006.

[16] P. Nikander and J. Laganier. *Host Identity Protocol (HIP) Domain Name System (DNS) Extensions*. Internet-Draft draft-ietf-hip-dns-06, Work in Progress. February 2006.

[17] T. Henderson. *End-Host Mobility and Multihoming with the Host Identity Protocol*. Internet-Draft draft-ietf-hip-mm-03, Work in Progress. February 2006.

[18] J. Laganier and L. Eggert. *Host Identity Protocol (HIP) Rendezvous Extension*. Internet-Draft draft-ietf-hip-rvs-04, Work in Progress. October 2005.

[19] J. Laganier, T. Koponen and L. Eggert. *Host Identity Protocol (HIP) Registration Extension*. Internet-Draft draft-ietf-hip-registration-01, Work in Progress. December 2005.

[20] P. Nikander and J. Melen. *A Bound End-to-End Tunnel (BEET) mode for ESP*. Internet-Draft draft-nikander-esp-beet-mode-05, Work in Progress. February 2006.

List of Figures

Appendix A

NAT Examples

In this appendix, some examples are presented to describe the various behaviors of NATs according to their type. In these examples, several UDP packets with various IP addresses and port numbers, are transmitted and are then either properly forwarded or discarded by NATs. The different types of NATs are treated in the following order :

- Full Cone NAT, in Figure A.1,
- Restricted Cone NAT, in Figure A.2,
- Port Restricted Cone NAT, in Figure A.3,
- Symmetric NAT, in Figure A.4.

In the following figures, *S* designates the source IP address and UDP port number of the transmitted packets. And *D* designates the destination IP address and UDP port number of these packets.

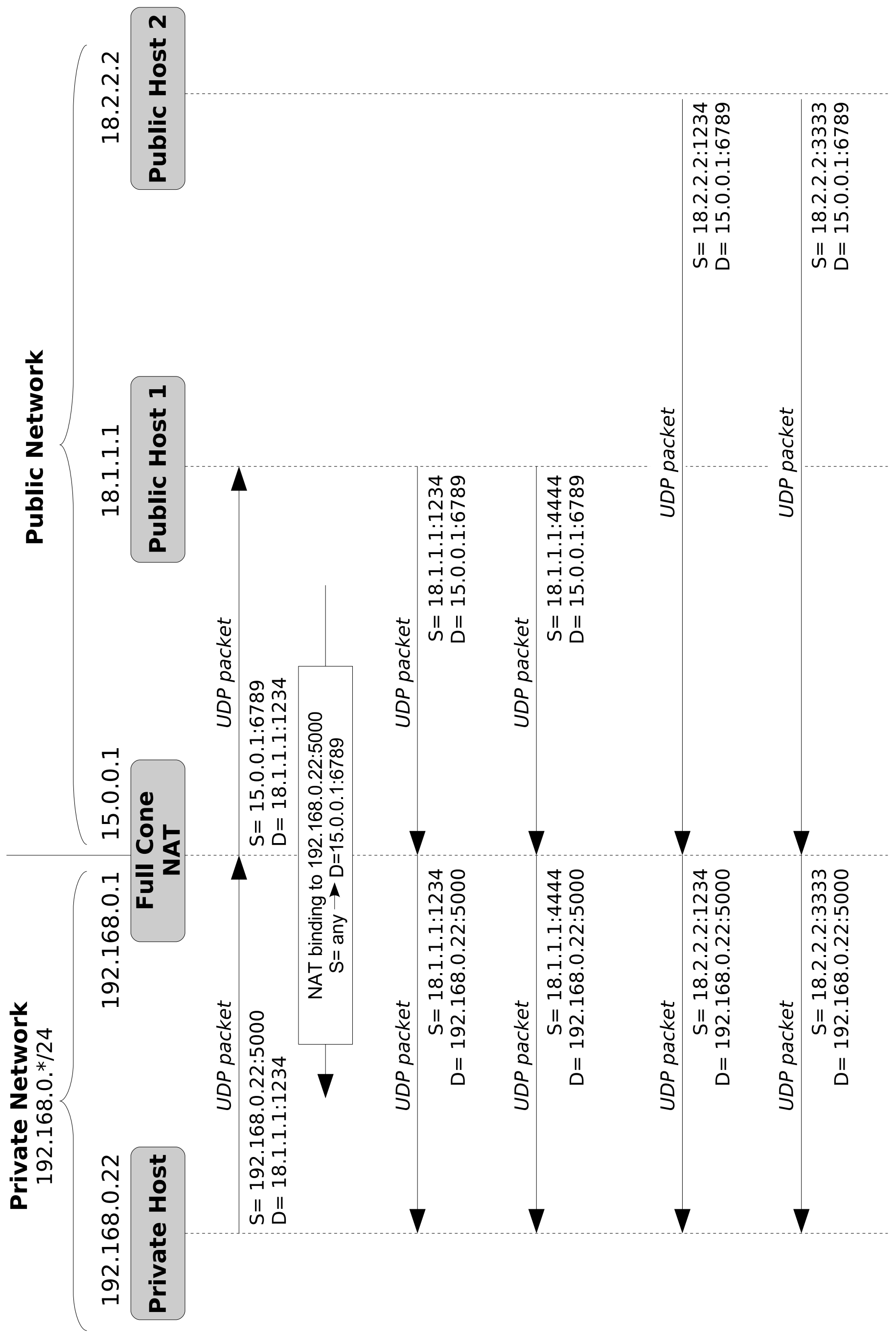

Figure A.1: Example of behavior of a full cone NAT.

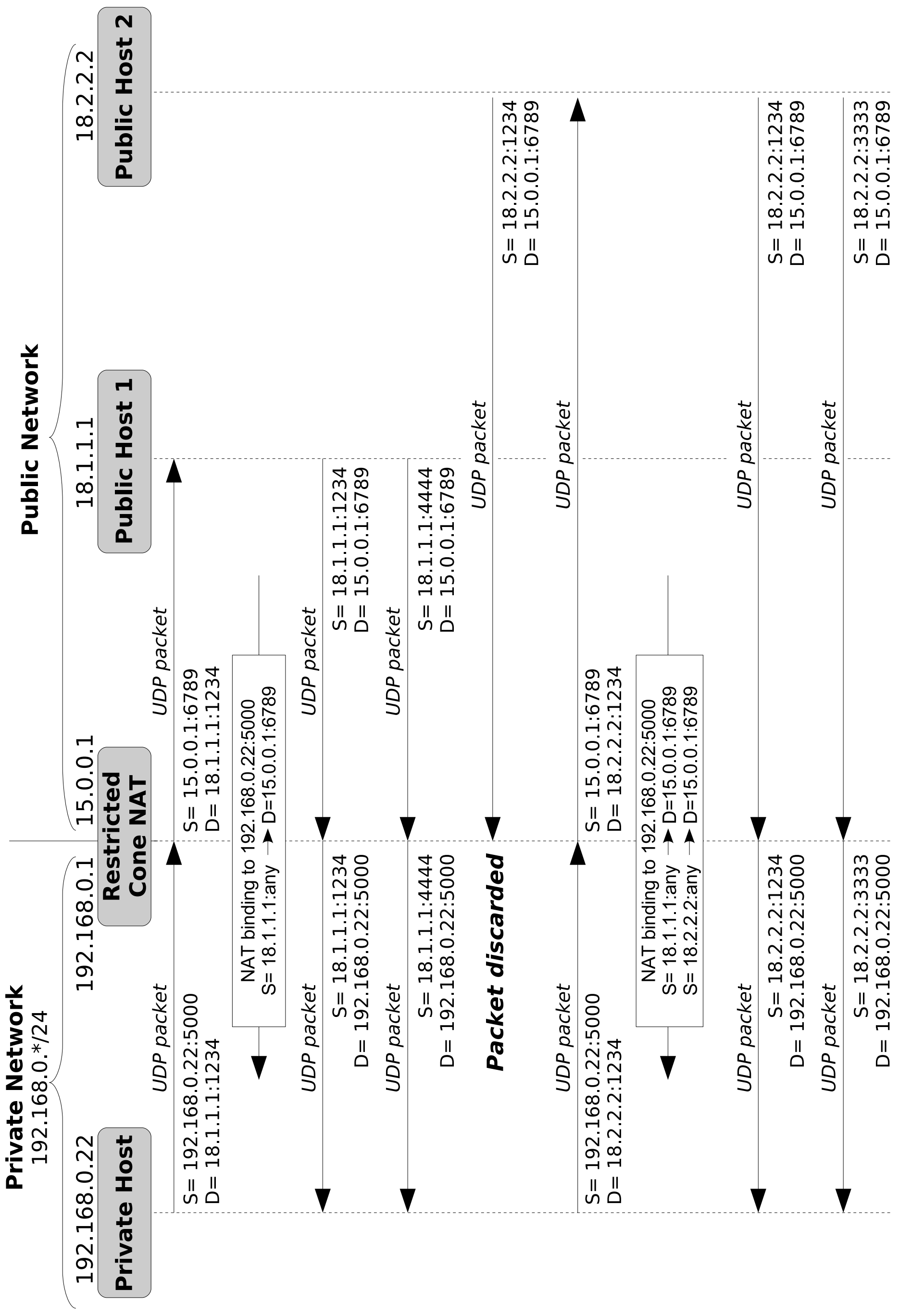

Figure A.2: Example of behavior of a restricted cone NAT.

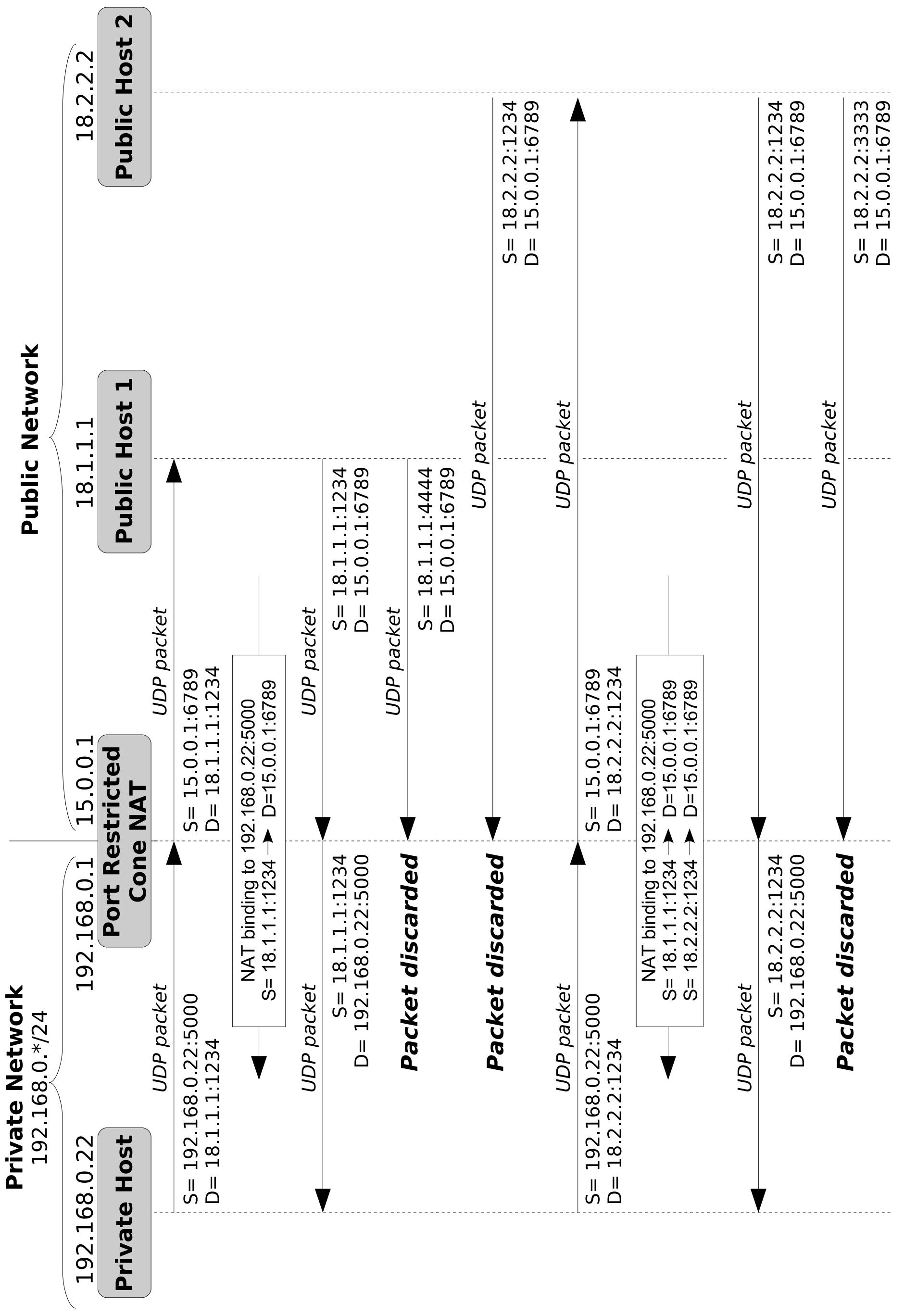

Figure A.3: Example of behavior of a port restricted cone NAT.

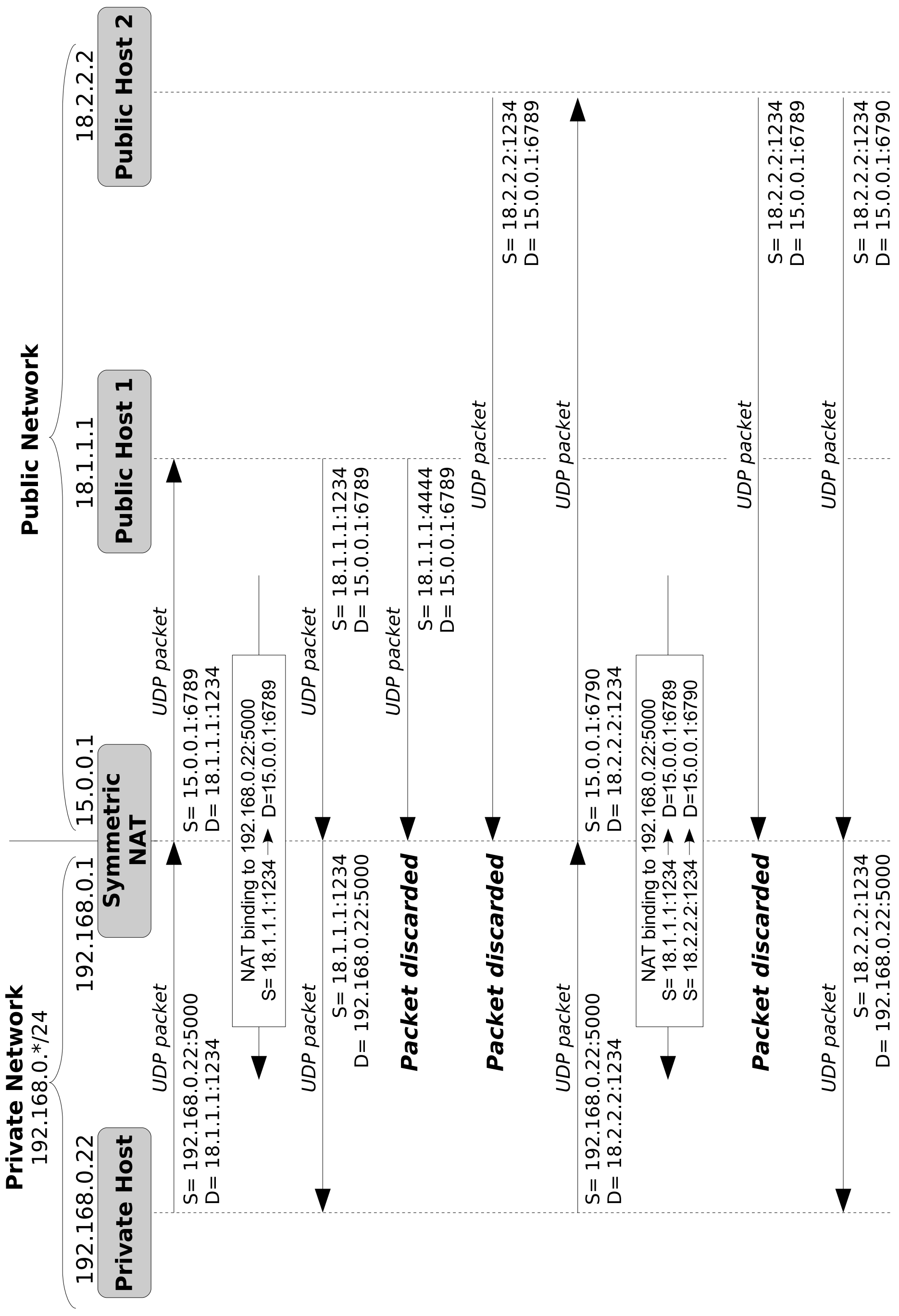

Figure A.4: Example of behavior of a symmetric NAT.