HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Electrical and Communications Engineering

Patrik Salmela

Host Identity Protocol proxy in a 3G system

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Technology

Kirkkonummi, Finland, 11th February 2005

Supervisor: Adjunct Professor (docent) Pekka Nikander
Instructor: Master of Science Petri Jokela

HELSINKI UNIVERSITY OF TECHNOLOGY **ABSTRACT OF THE MASTER'S THESIS**

Author: Patrik Salmela

Name of the thesis: Host Identity Protocol proxy in a 3G system

Date: 11.02.2005 **Number of pages:** 101

Faculty: Department of Electrical and Communications Engineering

Professorship: Telecommunication Software **Code:** T-110

Supervisor: Adjunct Professor (docent) Pekka Nikander

Instructor: Master of Science Petri Jokela

Concurrently with the Internet becoming increasingly popular, also the security of it has drawn much attention. This is partly a consequence of the Internet being used for business purposes. People are accessing the Internet using varying equipment, including mobile telephones. However, Internet connections from mobile telephones are not protected in the Internet, the security ends on the border of the telephone network.

The thesis examines how users of a 3G system can be provided with the services offered by a new protocol, the Host Identity Protocol (HIP). The main benefit provided by HIP to 3G users is security. The optimal method for providing HIP services to mobile telephones would be that all telephones were HIP enabled. However, this is not feasible since enabling HIP in a host requires that the IP-stack of the host is updated. The method presented in the thesis is to have a HIP proxy located in the 3G network. The proxy provides HIP for part of the connection, i.e. between the proxy and the connection endpoint somewhere in the Internet.

A prototype is constructed to prove the concept of a HIP proxy. However, the implementation is not done for a 3G system, but for a computer network environment.

Keywords: HIP, proxy, identity, IPsec, 3G, FreeBSD

HELSINGFORS TEKNISKA HÖGSKOLA **SAMMANFATTNING AV DIPLOMARBETET**

Författare: Patrik Salmela

Diplomarbetets titel: Host Identity Protocol mellanserver i ett 3G system

Datum: 11.02.2005 **Antalet sidor:** 101

Avdelning: Avdelningen för Elektro- och Telekommunikationsteknik

Professur: Telekommunikationsprogramvara **Kod:** T-110

Övervakare: Docent Pekka Nikander

Handledare: Diplom Ingenjör Petri Jokela

Samtidigt som Internet blivit allt vanligare har man också blivit allt månare om säkerheten i Internet. Detta beror delvis på att Internet används för kommersiellt bruk. Man kan koppla upp sig till Internet med olika metoder, bland annat med en mobiltelefon. När man kopplar upp sig till Internet från en mobiltelefon är förbindelsen skyddad i telefonnätet, men i Internet är trafiken oskyddad.

I det här arbetet undersöks hur man skulle kunna utnyttja egenskaperna av ett nytt protokoll, Host Identity Protokollet (HIP), för förbindelser från 3G-nät. Den viktigaste egenskapen HIP kan erbjuda användare av 3G är informationssäkerhet. En optimal lösning vore att varje mobiltelefon var HIP-kapabel. Det är dock inte en rimlig lösning eftersom det skulle kräva att varje mobiletelfon blev uppdaterad till att stöda HIP. Lösningen som presenteras i det här arbetet baserar sig på att man har en HIP-mellanserver i 3G-nätet. Med hjälp av mellanservern kan man använda HIP under en del av förbindelsen, nämligen mellan mellanservern och förbindelsens endpunkt i Internet.

För att bevisa att HIP-mellanserver konceptet är fungerande i praktiken, implementeras en prototyp av mellanservern. Den är dock inte gjord för ett 3G-nät utan för ett datanät.

Nyckelord: HIP, mellanserver, identitet, IPsec, 3G, FreeBSD

Preface

This thesis was written at Oy LM Ericsson Ab, Finland. The subject of the thesis and the opinions stated in this document are solely my own and do not reflect the official Ericsson policy.

I would like to thank Ericsson for giving me the opportunity to do this work. My gratitude also goes out to my superiors who let me focus on the task at hand, and who together with my co-workers created an innovative working environment.

I wish to thank my supervisor, Pekka Nikander, for his guidance. I'm very grateful for the valuable advice and comments.

I would especially like to thank my instructor, Petri Jokela, for the ideas and insight he provided, and for helping me form this document; giving feedback even for the umpteenth version of a chapter.

I would also like to thank Jan Melén for his patience and all the help with practical issues to get the implementation environment working to satisfaction. Furthermore I would like to thank Jukka Ylitalo for helping me getting started with the implementation, and Jorma Wall for helping me remove the rough edges of this document.

Finally, I would like to thank my family and friends for their support, encouragement, and care. Thank You.

11th February, 2005 in Kirkkonummi, Finland

Patrik Salmela

Table of Contents

List of Figures

List of Tables

Abbreviations and Concepts

2G	Second Generation
2.5G	Second and a half Generation
3G	Third Generation
3GPP	Third Generation Partnership Project
3GSM	3 Global Systems for Mobile communications (Third Generation services delivered on an evolved core GSM network)
AH	Authentication Header
AId	Association Identity
AUC	Authentication Centre
BOF	Birds of a Feather
BOS	HIP Bootstrap (packet)
BS	Base Station
BSC	Base Station Controller
BTS	Base Transceiver Station
CA	Certificate Authority
CDMA2000	Code Division Multiple Access 2000
CER	HIP Certificate (packet)
DNS	Domain Name System
DoS	Denial of Service
EDGE	Enhanced Data Rates for GSM Evolution
EIR	Equipment Identity Register
ESD	End System Designator
ESP	IP Encapsulating Security Payload
FA	Forwarding Agent
FARA	Forwarding directive, Association, and Rendezvous Architecture

FD	Forwarding Directive
FTP	File Transfer Protocol
GGSN	Gateway GPRS Support Node
GMSC	Gateway MSC
GPRS	General Packet Radio Service
GSE	Global-, Site- and End-System (designator)
GSM	Global Systems for Mobile communications
HI	Host Identity
HIP	Host Identity Protocol
HIT	Host Identity Tag
HLR	Home Location Register
HMAC	Keyed-Hash Message Authentication Code
I^3	Internet Indirection Infrastructure
ICMP	Internet Control Message Protocol
IEN	Internet Experiment Note
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Subscriber Identity
IP	Internet Protocol
IPng	IP Next Generation
IPsec	IP Security Protocol
IPv4	IP version 4
IPv5	IP version 5
IPv6	IP version 6
IRTF	Internet Research Task Force

kbps	Kilo bits per second
LSI	Local Scope Identifier
MAC	Media Access Control
Mbps	Mega bits per second
MS	Mobile Station
MSC	Mobile Switching Centre
NAT	Network Address Translation
NCP	Network Control Protocol
PC	Personal Computer
PDN	Packet Data Network
PDP	Packet Data Protocol
PGP	Pretty Good Privacy
PIN	Personal Identity Number
PKI	Public Key Infrastructure
PLMN	Public Land Mobile Network
PSTN	Public Switched Telephone Network
RFC	Request For Comments
RG	Routing Goop
RNC	Radio Network Controller
SA	Security Association
SADB	SA Database
SGSN	Serving GPRS Support Node
SIM	Subscriber Identity Module
SPI	Security Parameter Index
SSL	Secure Socket Layer
STP	Site Topology Partition

TCP	Transmission Control Protocol / Transfer Control Program
TFTP	Trivial File Transfer Protocol
TLS	Transport Layer Security
TLV	Type-Length-Value
TTL	Time To Live
UDP	User Datagram Protocol
UE	User Equipment
UMTS	Universal Mobile Telecommunications System
USIM	User Services Identity Module
VLR	Visitor Location Register

1 Introduction

Digital communication is widely in use all around the world. The number of mobile telephone and Internet users is growing, as well as the number of user equipment connected to these networks. The networks are also becoming increasingly more interconnected. Since the introduction of the General Packet Radio Service (GPRS) [1], it has been possible to establish packet switched data connections to the Internet from equipment connected to the mobile telephone network. These connections are logically always on, analogous to broadband Internet access. Already before GPRS was introduced it was possible to establish data connections to the Internet. These connections were circuit switched dial-up connections. However, like with Internet connections from personal computers, the trend is moving from dial-up connections to "always on" connections. From this point onwards, when talking about connections between mobile telephone networks and the Internet, the connections are assumed to be packet switched unless otherwise stated.

The two networks, the GPRS network and the Internet, are connected to each other via a node in the GPRS network; the Gateway GPRS Support Node (GGSN). The GGSN is also used by third generation mobile telephone (3G) systems such as Universal Mobile Telecommunications Systems (UMTS) [2]. The function of a GGSN is to work as a router between the mobile telephone network and the Internet. This is shown in Figure 1 below.

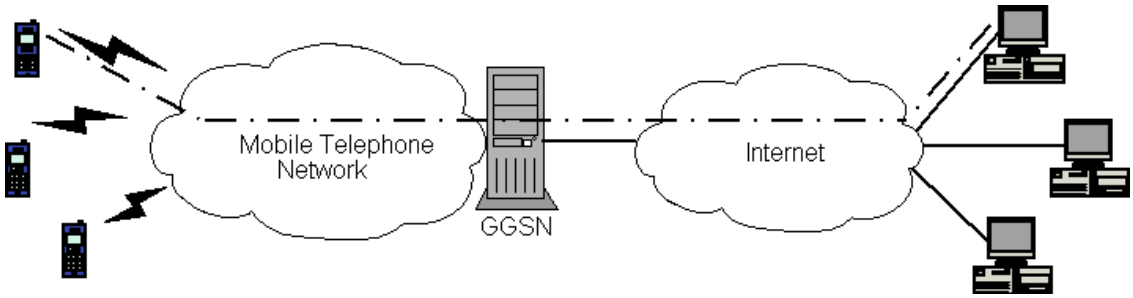

Figure 1: Mobile Telephone Network connected to the Internet via GGSN

When a mobile telephone is communicating with nodes in the Internet it needs to use addresses that the Internet can handle. If only IPv4 [3] addresses would be used for this purpose, the already heavily utilized IPv4 address space would even more quickly need replacement. The problem of the small address space of IPv4 has been acknowledged [4], and IPv6 [5] will in time replace the 32-bit address of IPv4 with a 128-bit address. A 128-bit address is considered to supply a sufficient address space for future needs.

The current Internet architecture is based on an over 20-year-old IPv4 protocol [6]. Since those days much has changed. There are now new technologies and needs - the solutions of yesterday might not be enough to fulfill the current needs. The problem of the small address space of IPv4 is one thing that needs attention. Another area that needs improvement is security. The initial design of the Internet was not focused on security. All data was sent in plaintext so anyone who wanted, and had the needed knowledge, could read the transmitted information. Since the introduction of services like on-line shopping security has become more important. Different solutions for making the communication secure have been developed, including e.g. PGP [7], TLS [8] and IPsec [9]. The Internet is still insecure, but now there are assorted tools for securing the communication over it.

Currently the IP address has two functions; it is used to route traffic to the destination node and at the same time it serves as the identifier of the node. The dual role of the IP address causes some problems. When a mobile node moves to another location in the network topology the IP address of the node changes. The consequence of this is that the information used to route packets to that node is changed. But, as the IP address also serves as the identifier, the identifier is also changed. This means that the same node would have different identifiers depending on where it is positioned in the network. To be useful the identifier to be should remain the same regardless of where the node is located. Methods for solving the ambiguity problem of the IP address have been presented. There are solutions that attempt to solve the problem using resources and technologies we have now. An example of this is Mobile IP [10], which tries to fix the problem by assigning multiple IP addresses to a node. This is more like bypassing the problem instead of repairing it. There are also solutions that instead try to separate

the identifiers from the routing information by modifying the current architecture. One such proposal is the Host Identity Protocol (HIP) [11].

1.1 A new Internet architecture

The ambiguity of the IP address is a known problem. This problem can be solved by changing the way the IP address is used. This means a new Internet architecture. Alternatively some other new, yet to be discovered, solution has to be introduced. There are some solutions that attempt to change the current Internet architecture: PeerNet [12] is a solution that abandons the IP address and uses an own addressing system. Another solution attempt is FARA [13], which does not go into implementation details but instead defines an abstract framework that can be used to derive architectures. Other work includes the GSE proposal for IPv6 [14], I^3 [15] and HIP. These will all be looked at in Chapter 2.

HIP, the Host Identity Protocol, is a rather new concept which tackles some of the concerns regarding the Internet Protocol. HIP separates the identifier from the locator and also provides security and mobility functionality. With HIP, the identifier of a node, known as the Host Identity (HI), remains constant regardless of node movements in the network. So even if the IP address changes when the node moves around in the network the HI will stay the same. Each HIP enabled host has an asymmetric key pair and the public key of this key-pair serves as the Host Identity. Besides serving as the identity of the host, the key-pair can be used to provide security. The security functionality of HIP is very much like that of IPsec. The difference is that with HIP the Internet Key Exchange (IKE) [16] is not needed. This is because HIP initiates a 4-way handshake that establishes a session key using the Diffie-Hellman procedure [17].

In the current Internet architecture packets are delivered to the node that is identified by the destination IP address of a packet. With the introduction of HIP, the destination of packets will instead be identified by Host Identities. Packets will still be routed with the help of an IP address, but when they arrive at the destination host they are

processed based on the Host Identity. This results in that a modified IP-stack is needed on the end-hosts using HIP. The fact that a modified IP-stack is needed makes deployment of HIP quite challenging, especially in a network as big as the Internet with millions of potential HIP hosts.

1.2 Goal

Connections between a mobile telephone network and the Internet introduce new security concerns. Since it is a private network, the mobile telephone network can, from the security perspective, be considered to be secure. However, the connection between the edge of the mobile telephone network and the node on the Internet is not secure. Adding security to that part of the connection would be a big improvement to the overall security of connections between the Internet and the mobile telephone network.

In this thesis the possibilities to get mobile telephones to take advantage of the services provided by HIP are investigated. It is not feasible to expect that all new mobile telephones would have the modified IP-stack required by HIP. Nor is it feasible to expect that all old mobile telephones would be updated to support HIP. To benefit from the features of HIP another approach is needed. HIP could be used to provide security between the gateway nodes of the mobile telephone network, i.e. GGSN, and the nodes in the Internet. The gateway node could implement HIP and thus handle connections between a legacy mobile telephone using IP and a HIP enabled peer node. In this case the mobile telephone would believe that it is communicating with the node in the Internet using IP directly while the node in the Internet sees a HIP protected connection. This can be achieved by implementing a HIP proxy in the GGSN.

The primary goal is to implement a HIP proxy that will function in the following way: When the HIP proxy receives IP traffic destined at a HIP enabled node it needs to forward the packets using a HIP association established between itself and the HIP node. This involves performing security functions on the packet according to the HIP

association and sending the modified packet to its true destination. Likewise, when the HIP proxy receives traffic over the HIP association it needs to perform the reverse security functions on the packets. After this, the modified packets can be sent as IP packets to their true destination, in this case the mobile telephone. The true endpoints of this communication should not be aware of the changes done by the proxy. This results in part of the communication being done over a HIP association and the other part using "regular" IP. This is depicted in Figure 2.

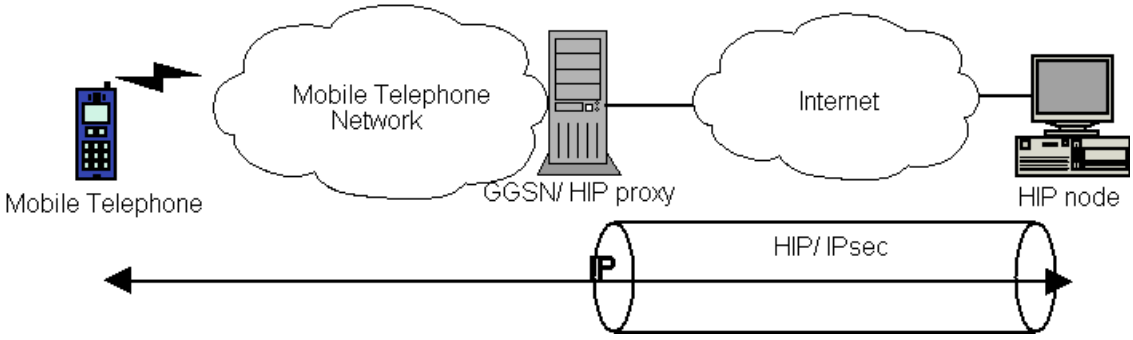

Figure 2: A HIP proxy in a GGSN node

The rest of this thesis is structured as follows; Chapter 2 presents some background for the work, and the architecture and security of IP networks and mobile telephone networks are studied. The HIP protocol is presented along with some other attempts of changing the Internet architecture.

Chapter 3 deals with the problem statement; "Why do we need a HIP proxy?". The chapter also looks at how HIP should be integrated into the 3G system UMTS.

Chapter 4 explains the functionality of HIP and the HIP proxy in more detail. Also the requirements for the proxy imposed by the GGSN are looked at. In Chapter 5 the design and implementation of the HIP proxy is presented.

In Chapter 6 the work is evaluated and the results are analyzed. Also other possible solutions are looked at. Finally, in Chapter 7 the conclusions are presented.

2 Background

This chapter will present background information related to the thesis. This includes the TCP/IP architecture and security issues associated with it. The relevant security terminology will be explained. The problems and shortcomings of the architecture are discussed and some attempts to alter this architecture are looked at. Finally, the mobile telephone networks and their structure are presented along with the used security methods.

2.1 History of IP network protocols

The Internet is full of resources about the history of itself. Some, such as "History of the Internet and Web" [18] written by Anthony Anderberg, begin as early as 700 years BC telling about how, in ancient Greece, they used homing pigeons to carry messages. But most of the timelines, such as [6], begin in the 20th century.

In 1961 the first paper on packet-switching theory [19] was presented. Less than 10 years later, in 1970, hosts in ARPANET begun to use the first host to host protocol, the Network Control Protocol (NCP) [20]. After this the development was quite rapid, and in 1973 TCP was referenced for the first time in a paper [21]. The next year, 1974, TCP (at the time, Transfer Control Program) was specified in RFC-675 and a three-way handshake was taken into use. Over the next few years the TCP specification was improved and revised and in 1977, the first Internet, based on three networks using TCP (packet radio, ARPANET, SATNET), was demonstrated. Later the same year Jon Postel wrote an IEN [22] (older form of RFC) that discussed the idea of using a combination of a hop-to-hop protocol and an end-to-end protocol. In 1978 Postel and Vint Cerf together wrote IEN 21, which was the third TCP specification; TCP/IP started to take form. In 1979 and 1980 Postel continued to develop the specifications for TCP and IP. Then in 1981, IPv4, which is the primary Internet protocol used today, was defined in RFC-791. In 1983 ARPANET switched from NCP to IP.

Since 1981 and the introduction of IPv4, many shortcomings of it have been discovered; e.g. the limited address space, lack of security and the bad scalability of routing [23]. During the past years, solutions and tools have been developed that target most of these problems. One solution for many of the problems of IPv4 is IPv6. The roots of IPv6 go back to 1991 when IETF took on the challenge of trying to increase the IP address space. It resulted in moving from the 32-bit addresses in IPv4 to 128-bit addresses in IPv6. With a bigger address size the IP protocol structure had to be changed and that meant a new version of the IP protocol; the IPv6 (there was also an IPv5, but it was only used as an experimental protocol). As the protocol structure of IPv6 would be different than that of IPv4 it was decided that also other changes, e.g. more security functionality, could be considered if there was a need. This led to that the IPv6 protocol now has many new features and improvements compared to IPv4, not only the bigger address space.

2.1.1 TCP/IP architecture

The TCP/IP architecture can also be called the Internet architecture since the two protocols, TCP and IP, are the main protocols of the Internet protocol family. In Figure 3 [3] the Internet protocol hierarchy is depicted. It can be seen as consisting of four layers; the Application-, Transport-, Internetworking- and Network layers.

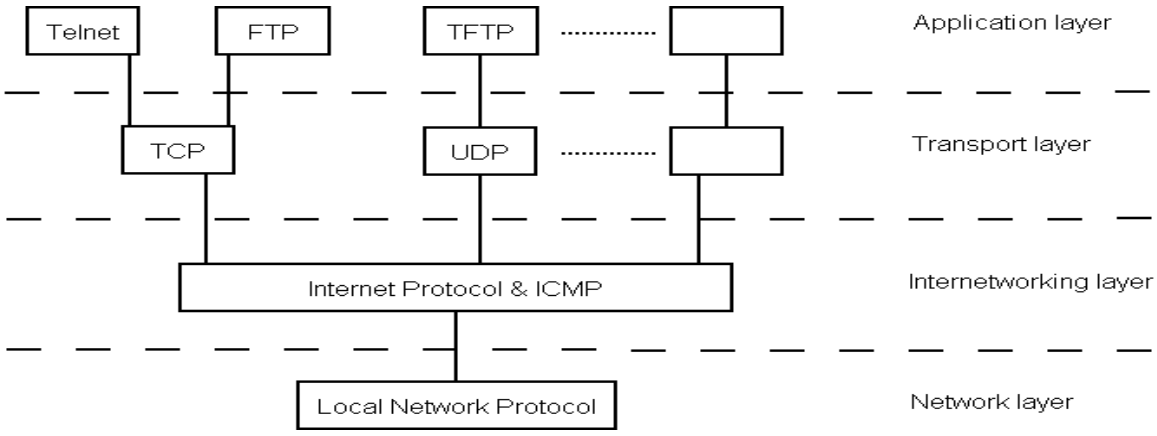

Figure 3: The Internet protocol hierarchy

On the application layer resides the application protocols such as Telnet and FTP. When, e.g., an FTP application is sending data over the Internet the data is first handled according to the application and the application protocol. The resulting data packets are passed to the transport layer where, in this case, a TCP header is attached to the packet. To be able to have different applications that at the same time use the same service, e.g. TCP, each application uses a 16-bit port number to mark packets belonging to the corresponding application.

The new packet, which consists of application data and a TCP header, is handed over to the internetworking layer where an IP header is attached to it. The IP header contains the IP address that is used to route the packet to the final destination. Finally the packet is passed to the Network layer where the local network protocol, which can be e.g. Ethernet [24], is applied to get the data to move through the network. During the journey of the packet through the Internet, the network protocol can change many times depending on the protocol used by the underlying network. When the packet reaches the destination node it is taken through the same layers, in opposite order, and the corresponding headers are removed until the data finally is delivered to the application.

2.1.1.1 Problems with the current architecture

The problems of the current architecture are very much centralized in one protocol, the Internet protocol. There are problems concerning its attributes as well as the definition of one of its primary units, the IP address. Most of the issues concerning the attributes of the current version of the Internet protocol, IPv4, are fixed by IPv6, which in the future will replace IPv4. However, the migration to IPv6 will be challenging since it will affect all nodes in the Internet. A comparison of IPv4 and IPv6 will be presented in Section 2.1.1.2.

One of the most acute problems of IPv4 is the small address space. With a 32-bit address field only a little over 4 billion (2^32) unique addresses can be defined. However, the utilization of the address space is not going to reach 100%, so around 4

billion addresses is only a theoretical maximum value. One of the reasons why the size of the address space of IPv4 is considered to be inadequate is because the Internet has become very popular. This has lead to that the amount of personal computers connected to the Internet is rapidly increasing. But also mobile devices, such as mobile telephones, can nowadays be attached to the Internet and thus need an own address.

The reason why the address space is not going to be fully utilized is because the address allocation is not very flexible. Sites are assigned address blocks depending on their needs. However, there are only three different address classes, called A, B and C, and they provide address blocks for three different network sizes [3]. The class an address belongs to can be identified by the high order bits of the address. The address is further divided into a network and a host part. The network part of a class A address is only 7 bits so there can only be 128 (2^7) networks using class A addresses (actually 126, since 0 and 127 are reserved [23]). However, the host part of a class A address is 24 bits, which allows for about 16 million (2^24) hosts to be connected to a class A network. Classes B and C have bigger network parts and smaller host parts, allowing for more networks with a smaller amount of host compared to class A networks. The fact that the address classes have predefined sizes limits the utilization of the addresses. If a network does not exactly require the amount of address provided by one of the classes some of the assigned addresses will not be used.

Since the address field of IPv4 is only 32-bits there is not much room for addressing hierarchy. This results in very big routing tables, which makes routing very demanding for the routers. There are also many features, such as security, that people have recognized that they need, which are not implemented in IPv4. Many of these requirements have been addressed with tools and methods developed to fit the needs so that it is possible to have these features in an IPv4 based network. In IPv6 many of these features are a part of the protocol or easily supported by it.

Another issue, even if it does not affect the functionality directly, is the definition of the IP address. The IP address, regardless of version, serves two functions; it is used to route packets in the Internet but it also serves as the identifier of the node. The problem

is that these two functions have, by definition, different attributes. The routing information is used to route packets to the destination node. When a node changes position in the network the routing information for the node needs to change to get packets routed to it correctly. This is satisfied by how a node gets a new IP address when connected to a new position in the network. However, the other functionality of the IP address, the node identification, does not conform to its definition. The identifier of a node should be assigned once and then stay constant regardless of the position in the network. The IP address however changes when the node changes its topological location. Thus it cannot be directly used as an identifier. The IP address should only be used as routing information.

A solution to this problem would be to introduce a new per host variable that would serve as an identifier of the host. This issue has generated different solution possibilities, such as the GSE proposal for IPv6 and HIP. These, and other solutions, will be presented later in Sections 2.1.2 - 2.1.4.

2.1.1.2 IPv4 vs. IPv6

The most important difference between the two versions of the Internet Protocol is the size of the IP address. With a 128-bit address it has been estimated that the address space allows for around 1500 addresses per square foot (approx. 30cm*30cm) on the earths surface [23]. This is based on a pessimistic estimate of address utilization. From this number it is easy to conclude that the problem of a too small address space should be solved with IPv6. The big address also makes address aggregation easier. With 128 bits it is possible to introduce many layers of hierarchy in the address. This will reduce the size of routing tables and thus ease the load of the routers, which is a concern in the current Internet.

The fact that an IPv6 address is four times bigger than an IPv4 address could cause concern that the packet overhead would get too big. However, as the header structure is not the same for the two versions, the header size of IPv6 (40 bytes) is in fact only

twice as big as that of IPv4 without any options (20 bytes). The two header structures are presented in Figure 4.

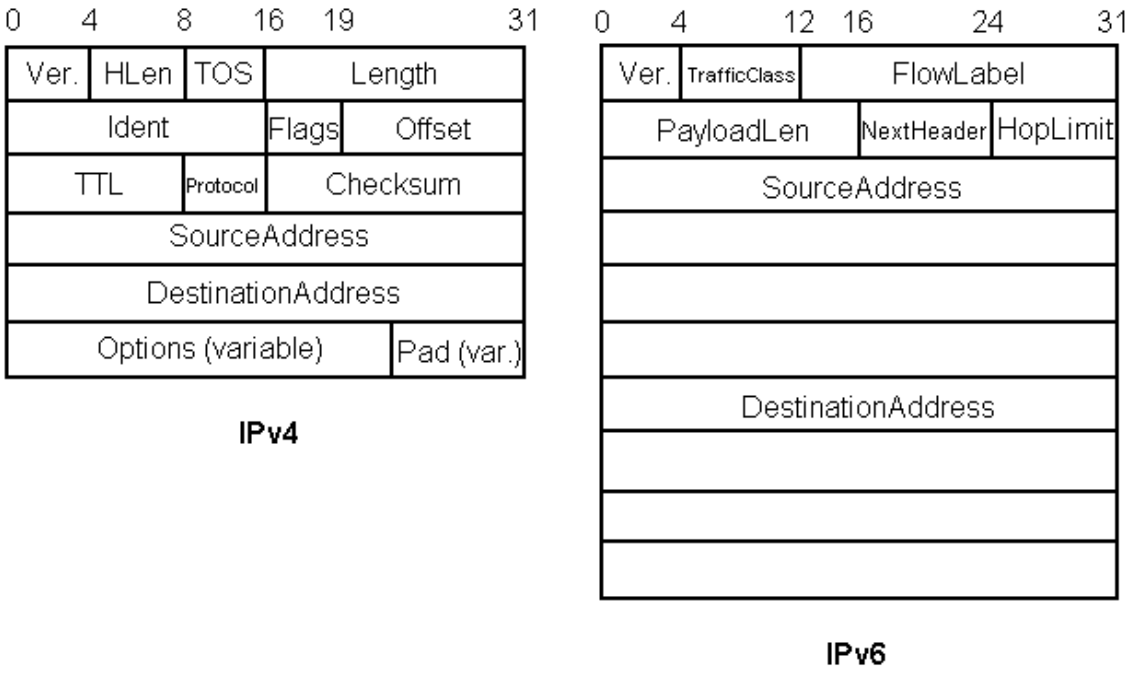

Figure 4: Header structures of IPv4 and IPv6

In both the IPv4 and the IPv6 header there are fields for source and destination IP addresses. Also a field for the Version information is present in both headers, it specifies the version of the used protocol. The fields TrafficClass and FlowLabel of the IPv6 header are concerned with quality of service. The HopLimit in the IPv6 header equals the Time To Live (TTL) field of the IPv4 header. From Figure 4 it can be seen that in the IPv6 header there are not any variable length fields, thus the header is always the same size. This makes processing of the IPv6 headers easier. The purpose that the variable length fields (mainly Options) in the IPv4 header serve, are in the IPv6 header handled by extension headers and the NextHeader field.

Some of the options that can be selected contain information meant for the routers. When an option is present it is indicated by the NextHeader field which describes the extension header that follows. Thus it is easy for the router to just check the NextHeader field to see if the option is of any interest for it. There are also rules in

which order the extension headers should be present. This also helps the routers to parse the required information. Amongst the extensions are options concerning routing and security. In IPv4, a router always has to parse the whole header of a received packet to check if the options are meant for it. Thus including options in IPv4 headers strains routers significantly more compared to IPv6

2.1.2 The GSE proposal for IPv6

The letters GSE come from the concepts Global-, Site- and End-System Designator. GSE was developed with hopes that it could be used as the new addressing architecture in IPv6. It is an attempt to separate the locator from the identifier inside the IP address. However, GSE has some disadvantages, e.g. solutions for multicast and mobility have not been considered. The main concern, which led to the development of GSE, was the cost of renumbering when the service provider of a site was switched. In 1997 the IPng (IP Next Generation, also known as IPv6) Working Group held a meeting where the GSE proposal was examined [14].

The idea of GSE is that the IP (IPv6) address is divided into three parts with corresponding tasks indicated by the letters in the name. The global part is 50 bits long and called “Routing Goop” (RG). It gives the position where the site, to which the host belongs, is connected to the Internet. The site part, which is called “Site Topology Partition” (STP), is 14 bits long and indicates the link inside the site where the host is located. The last 8 bytes of an IPv6 address serve as the identifier of the host, it is called the “End System Designator” (ESD). Thus the first 8 bytes of the address would be used to route packets, and correspondingly the last 8 bytes of the address would be used for identification of the node. The GSE partitioning of the IPv6 address is illustrated in Figure 5 [14].

Figure 5: GSE partitioning of the IPv6 address

When using GSE the network topology is required to have a tree-like structure so that the sites are positioned as leafs in that tree. The Routing Goop is used in n-bit sequences to route the packets downwards in the tree from bigger topology structures to smaller ones until reaching the target site. The largest topology structures, which reach the highest in the tree structure, are called “Large Structures”. 13 bits of the Routing Goop is designated for selecting between them. The 14-bit STP is used for routing inside the designated site. The 8-byte ESD serves as an interface identifier. To get it to uniquely identify the interface the plan was to derive it from the Media Access Control (MAC) address of the corresponding interface. This means that a node keeps its ESD even if it switches service provider, thus it truly serves as an identifier.

Nodes belonging to a site do not know the RG of the site, instead they use a site-local RG address. This way, if the site changes the service provider, the nodes inside the site can continue to communicate with each other without reconfiguration. When a node communicates with nodes outside of the site, the border routers of the site change the RG part of the source address in the outgoing packets to match the true RG address of the site. Without this modification the reply packets would not be routed back correctly. Similarly the destination addresses are rewritten for incoming packets.

After studying the GSE proposal the IPng Working Group did not however rule in its favor. Even if the overloading of the IP address with tasks of both identifier and locator is not a 100% good solution, the GSE has some problems that were considered to be too critical. The main problems with GSE are related to security. With GSE some forms of attacks are easier to perform than with standard IP. The lack of any authentication of the identity of endpoints results in opportunities to steal identities. This in turn opens up possibilities for both denial of service (DoS) attacks and

connection hijackings. Another big problem with the GSE proposal is that there is not a straightforward way of mapping from identifier to locator.

2.1.3 The HIP proposal

Even if the GSE proposal was not a success, the idea of trying to separate the identifier from the locator has not died with it. The Host Identity Protocol (HIP), which has a major role in this thesis, is one proposal that targets the same separation problem. HIP was born in 1999 [25], and in 2001 an IETF working group was suggested but it did not yet happen. However, at the 58th IETF meeting, in Minneapolis in November of 2003, a HIP BOF meeting was held and it was decided to form an IETF working group and an IRTF research group [26].

HIP separates the identifier from the locator with the help of a new entity, the Host Identity (HI). The IP address is still used as the locator while the HI serves as the identifier. The HI is the public key of an asymmetric key-pair. However, because of its length it is not feasible to use it during actual communication. Instead a 128-bit hash of the HI, called the Host Identity Tag (HIT), is used. The length of the HIT allows it to be used instead of an IPv6 address at higher layers. In IPv4 architectures a 32-bit hash of the HI called the Local Scope Identifier (LSI) is used. The size of the LSI makes collisions quite probable, thus it can be considered to be unique only in a local scope. In a HIP capable node, when using HIP, the applications use the HIT (or LSI in IPv4) as the destination for the packets. The IP address is hidden from the applications and a translation from HIT to IP address must be made at some point in the IP-stack. To handle this translation a new layer is added to the network architecture presented in Figure 3. In Figure 6 [25] the new architecture, with the new Host Identity layer, is presented. In all layers above the Host Identity layer, a HIT is used instead of an IP address to represent the host. At the Host Identity layer the HIT is translated into an IP address for correct routing in the network (or IP address to HIT when receiving packets). In all layers below the Host Identity layer everything works as in the current

architecture. A node learns of the HIT of a peer in the same manner as it would a normal IP address, e.g. via DNS.

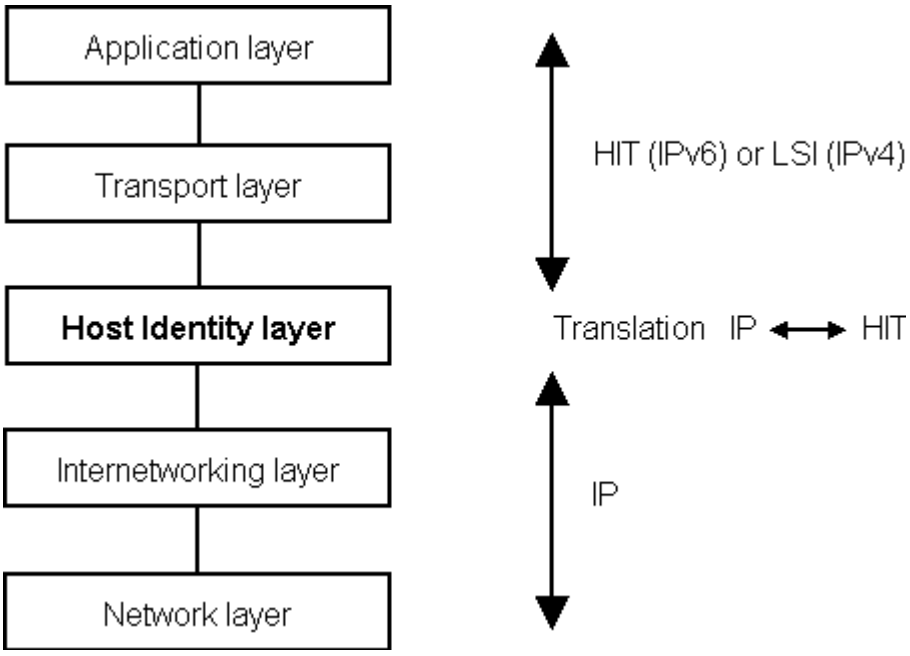

Figure 6: The layering architecture when using HIP

Before two HIP nodes can communicate with each other using HIP they perform a 4-way handshake called the HIP base exchange. During the base exchange they create a session key, using the Diffie-Hellman procedure [17], to be used in IPsec Encapsulating Security Payload (ESP) [26] Security Associations (SA) [9]. Instead of binding the SAs to IP addresses as the current IPsec defines, the SAs are bound to HITs. Because of this, even if one of the nodes moves and gets a new IP address, the SAs stay valid.

The HIP architecture allows the location information to be changed during communication. The HIP mobility management is achieved by letting the mobile node send an update message to its peer, informing the peer of its new IP address. Using this received address the peer node can update its tables of IP to HIT translations, and thus get them to reflect the current situation of the mobile node. The mobility provided by HIP is one of the big benefits of having a separate locator and identifier; connected nodes use the host identity (HI) to identify the peer and when one of the nodes moves it

updates its location to the other peer. The connection is maintained because it still uses the HI for identification. The reason for this is that you are still sending messages to the identity, only the way the messages travel to it changes. From an end-user perspective this is very much like mobile telephones; you call a friend and the calls reach him/her regardless of location (as long as there is coverage). You and the other party can move freely while communicating and the channel will stay open.

Mobility and security, along with separation of identifier from locator are the main strongpoint of HIP compared to regular IP. As already said, the 4-way handshake creates ESP SAs for the connection. They will provide the connection with security in the form of confidentiality, limited traffic flow confidentiality, data origin authentication, connectionless integrity and an anti-replay service [27]. HIP architecture also supports multi-homing as a natural extension. A node is multi-homed when it has more than one interface simultaneously attached to the network and it can use any of them. Because of this there are different addresses that can be used to route packets to the node. With HIP, packets are always sent to the identifier, the HIT. When a node is multi-homed the packets can be sent to it through different addresses. This can e.g. be useful when there is congestion on one of the paths to the node. In Chapter 4 there will be a more detailed description of HIP communication.

2.1.4 Other proposals for separating the identifier from the locator

The GSE and HIP proposals are not the only attempts to separate the identifier from the locator. In this section three other solutions, FARA, Internet Indirection Infrastructure (I^3) and PeerNet, are presented.

2.1.4.1 Forwarding directive, Association, and Rendezvous Architecture (FARA)

FARA [13] is not a concrete solution but a framework, a collection of models and ideas, which can be used to engineer a new architecture. The FARA architecture model is divided into two layers. It has an upper layer in which the communicating entities

and the communication endpoints reside, and a lower layer that handles communication through connectionless packet forwarding.

FARA uses, without defining the format, Forwarding Directives (FD) to navigate packets through the networks. Using FDs the packets are not delivered to the destination *node* but to a destination *entity*. The destination entity could be thought of as an application, so connections can be considered to be between two applications. The communication link between two entities is statefull and is called an association. Associations of an entity are identified by a locally unique association ID (AId). The AId is defined per entity. If an entity moves, association IDs remain the same but the FD describing the routing information has to change. This is the ground for mobility in FARA, and it is based on how FARA separates identifier from locator. HIP could be used as part of a FARA architecture, e.g. for setting up secure associations [13].

2.1.4.2 Internet Indirection Infrastructure (I^3)

I^3 [15] introduces a new set of nodes to the current Internet, the I^3 servers. Nodes that want to receive packets in the I^3 system need to register their identity at one of the I^3 servers. The registration of the identity is called inserting a trigger. The trigger is a (ID, IP) pair, where the ID is the identity of the receiver and IP is the IP address of the receiver. The trigger has to be updated periodically or it will be removed from the I^3 server. In I^3 packets are sent to an identity. The packet travels the Internet searching for the I^3 server where the identity is registered. When the packet reaches the I^3 server where the identity is registered that server replaces the identity from the destination field with the IP address that is registered with the identity. The packet is from there then delivered to the destination.

The I^3 system allows multiple nodes to register with the same identity. A packet destined to the identity is delivered to all IP addresses that have been registered with the identity. This provides a multicast property for I^3. When a node moves around in the network it updates its trigger at the I^3 server to keep the correct IP address mapping. Thus the I^3 system also supports mobility. Because of the rendezvous architecture of I^3,

packets almost never take the shortest route to the destination node. This introduces delay to the system compared to regular IP traffic. There are also some security considerations, e.g., as many nodes can trigger the same identity eavesdropping is made very easy. Cryptography and private triggers have been presented as a solution for security problems. [15].

2.1.4.3 PeerNet

PeerNet [12], as the name might suggest, is based on peer-to-peer thinking. When a node wants to join the PeerNet network it contacts one of the nodes already in the network and asks it for an address. The node is assigned an address based on its location in the network and the address changes as the nodes changes position. However, the nodes identity stays the same regardless of position in the network. The addresses are selected as leafs of a binary tree. When a node joins PeerNet and asks a node for an address, if the asked node is not already positioned as a leaf in the binary tree, the asked node splits its address in two and assigns the two new addresses to itself and the asking node.

As an example, if the size of the address is 5 bits, and the asked node has the address $11\mathit{000}_{\text{bin}}$, the address is split into the two new addresses $111\mathit{00}_{\text{bin}}$ and $110\mathit{00}_{\text{bin}}$. The *italic* numbers indicate that the part of the address space is unassigned. Now the node with the address $111\mathit{00}_{\text{bin}}$ can assign addresses that begin with 111_{bin}. The other node, which has the address $110\mathit{00}_{\text{bin}}$, can assign addresses that begin with 110_{bin}. Routing in PeerNet is handled by distributed peer-to-peer routing, and each node in the network maintains some identity to address mappings. Which identity mappings the node maintains is related to the address of the node. The routing is performed bit-by-bit, descending the binary address tree. PeerNet is not a ready solution, e.g. security issues have not been addressed.

2.2 Security in IP networks

At the early days of the IP protocol and the Internet security was not a big concern. Now the Internet has grown and become more popular, the use of Internet banking and similar services has made security mandatory. In the following subsection some security related terms are explained to avoid misunderstandings. After that, some security methods will be presented.

2.2.1 Security related terms

Authentication
The process of verifying the identity of an object (e.g. process or human). Achieved by having the object prove its identity by presenting some data assumed to only be known by it, e.g. a password.

Authorization
The process of allowing or denying an object access to data. Often coupled to authentication; an object is first authenticated, then it is granted access to certain data based on the authenticated identity of the object.

Confidentiality
The property of data that it will only be available for authorized objects. Achieved e.g. by using cryptography to conceal the data. Thus only authorized objects, i.e. objects with the correct key, can access the data.

Cryptography
The technology and science of converting data to unintelligible form, and back, using mathematical algorithms.

Encryption
The task of converting readable data (plain text) to an unintelligible form (cipher text) using a mathematical algorithm and, usually, a key.

Decryption
The opposite of encryption. The task of converting data from an unintelligible form to a readable form using a mathematical algorithm and a key.

Key
A piece of information that is needed to perform encryption and decryption successfully.

Symmetric Cryptography
The form of cryptography where the same key is used both for encryption and decryption. This makes the key a secret element that should only be known by the objects that are authorized to access the protected data.

Asymmetric Cryptography/ Public-key Cryptography
In this form of cryptography there are different keys for encryption and decryption. Data encrypted with one of the keys from a key-pair can only be decrypted using the other key from the key-pair. Usually, with asymmetric cryptography, one of the keys is public while the other one is secret. Data is encrypted using the public key and decrypted using the private key. It can be done in the opposite way but then the data is accessible to all who have access to the public key. This method can be used to authenticate the holder of the private key.

Digital Signature
Digital signatures are used much like "ordinary" signatures. Data is signed to mark that the signer agrees on the signed data. The signed data can be produced by the signer but this is not a must. Data is signed by first calculating a hash over the data and then encrypting the hash with the private key of the signer. This way the receiver can check, by recalculating a hash over the data, if the data has been altered since it was sent. Signing the hash with the private key authenticates the sender.

Private Key
One of the two keys in an asymmetric key-pair. The private key should only be known by the objects that are authorized to access the secured data. Data encrypted with this key can be decrypted with the corresponding public key.

Public Key
The other key of the asymmetric key-pair. Just like the name suggests this key should be known to the public. To be more exact, it should be known to the public that might need to communicate securely with the holder of the corresponding private key. Data encrypted with this key can be decrypted with the corresponding private key.

2.2.2 The PKI and IKE

The Public Key Infrastructure (PKI) [28] is an architecture that can be used to exchange information securely over networks. To achieve this, PKI uses asymmetric encryption, public key databases and trust.

In PKI the communication security is achieved with the use of asymmetric encryption. To securely send data to an object the data is encrypted with the public key of the receiving object. Thus only the receiver, the owner of the corresponding private key, can open the encrypted data. Since symmetric encryption often is much faster than asymmetric encryption, the communication is regularly actually encrypted using a symmetric algorithm. The asymmetric keys are in this case only used for securely delivering the key for the symmetric encryption and for creating digital signatures.

In PKI, digital certificates are used to deliver public keys. Certificates hold information about the owner of the key-pair, to which the public key belongs. A certificate also has other relevant information regarding the certificate such as issuer and expiration date. The certificates are digitally signed by the issuer, the Certificate Authority (CA), which also has a certificate signed by its issuer and so on. The highest CAs in this CA hierarchy are called root CAs, they are assumed to be trusted. This way a user can check the issuer of a received certificate to see if it can be trusted. If not, the CA hierarchy can be climbed until a trusted CA is reached. When issuing certificates the issuer must verify the identity of the object requesting the certificate. Only after the CA is sure about the identity of an object is a certificate issued. To be useful, the certificates have to be easy to locate. This can be achieved by having them in a big

directory from where users can request them. If e.g. the private key gets compromised, or the certificate has to be removed because of some other reason, information of its removal will be posted on a certificate revocation list. This way, the validity of received certificates can also be verified by querying the revocation list. PKI is not a complete and problem free solution, some of its problems are discussed in [29].

The Internet Key Exchange (IKE) [16] is used for negotiating and setting up security properties for a connection between two nodes in a network. The result of this negotiation is called a security association (SA) [9]. An SA is a unidirectional logical connection. For a normal bidirectional connection between two nodes, two SAs are negotiated, one for each direction. For each SA there are defined keys for cryptographic functions, a security protocol and a security parameter index (SPI). The SPI is used to distinguish between multiple SAs in the same node. Packets are processed according to the information in the SAs.

IKE is performed in two phases; in Phase 1, using either Main Mode or the Aggressive Mode, a secure channel between the two communicating nodes is set up and the nodes are authenticated. The SAs created in Phase 1 are then used in Phase 2, the Quick Mode, to negotiate SAs for the connection between the two nodes. The SAs created in Phase 1 can be used to create multiple Phase 2 SAs for the two nodes. When the Phase 1 SAs expire the SAs created in Quick Mode will still continue to work until they expire. The authentication performed in Phase 1 can e.g. be done using certificates or pre-shared keys. The encryption keys are generated using the Diffie-Hellman procedure [17].

2.2.3 IPsec

IPsec [9] is a system that provides security services to communication channels for protocols from the internetworking layer (IP-layer) upwards. IPsec resides in the internetworking layer, thus it does not require any changes to upper layer applications using the IP-stack. The offered security services include access control, connectionless

integrity, data origin authentication, protection against replays, confidentiality and limited traffic flow confidentiality [9]. These services are offered with the help of two security protocols; the Encapsulating Security Payload (ESP) [27] and the Authentication Header (AH) [30] along with key management procedures such as IKE.

The AH protocol can be used to provide connectionless integrity, data origin authentication and protection against replays. ESP provides the same services as AH and in addition, confidentiality. However, AH provides a more thorough authentication as it also, contrary to ESP, covers some of the fields of the IP header. The two security protocols, AH and ESP, can be applied to a channel so that the channel only uses either of them or both of them. If both protocols are used ESP should be applied first and only then the AH protocol, this because of the difference in the authentication attributes of the protocols. Each protocol requires an own SA, so if both AH and ESP are applied then two security associations for each direction are needed.

When setting up an IPsec protected connection between two nodes IKE is usually used to negotiate session keys for the SA pair. During the SA creation the IPsec protocols are selected along with the security algorithms and the corresponding keys. All the information belonging to the SA, such as the used algorithms and keys, the SPI and destination IP address, are stored in a SA database (SADB). When handling packets the information in the SADB is referenced to perform the correct operations.

The IPsec protocols can function in two modes; Transport mode and Tunnel mode. Transport mode is used when the endpoints of the SAs are the communicating nodes themselves. In this case the IPsec header(s) are located between the IP header and the transport header. Alternatively, tunnel mode is used when either or both of the SA endpoints are not the end points of the connection. The SAs in a Tunnel mode connection can be between two security gateways as shown in Figure 7, or between a security gateway and one of the end point nodes of the connection. In Tunnel mode, when traveling the part of the connection that is protected by an SA, the packets have two IP headers as depicted in Figure 8. The outermost IP header has the IP address of the destination security gateway as the destination address. This can also can be the

connection endpoint depending on the setup. The inner IP header has the IP address of the connection endpoint in the destination field.

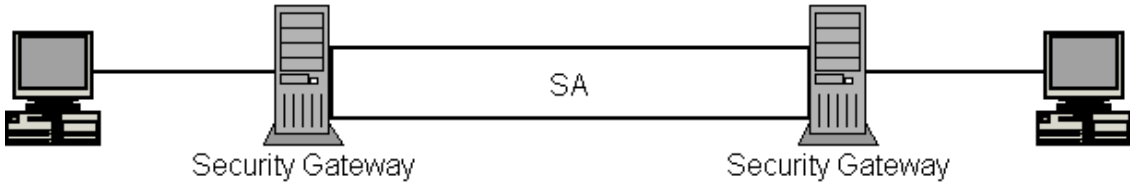

Figure 7: IPsec in Tunnel mode, SA between two security gateways

IP header Destination: Security Gateway	IPsec headers AH and/or ESP header	IP header Destination: Connection endpoint	TCP / UDP header	Data

Figure 8: Packet structure when using IPsec tunnel mode

2.2.4 Other IP network security methods/ protocols

Apart from what has been mentioned earlier there are also many other mechanisms and protocols that provide security in the Internet. Some of them are widely used, some rarely. In this subsection two of the more common methods, PGP and TLS, are described.

Pretty Good Privacy (PGP) [7] can be used for securing e-mail communication and files. This is achieved by using encryption, both symmetric and asymmetric. When securing e-mail communication the message itself is encrypted using symmetric encryption with a random 128-bit key. The used encryption key is sent in an encrypted form to the receiver so that the message can be decrypted. Asymmetric encryption is used for securing the key used to encrypt the message and for digital signatures. PGP uses public key encryption but it does not use the hierarchical model of CAs as PKI, in which the most trusted entity is highest up in the hierarchy. In PGP any user can act as a CA and sign public key certificates. As an example you can sign the public key certificate of your friend since you trust your friend. At some time someone who trusts

you might need to communicate with your friend. By recognizing your signature on your friend's certificate, this third party can trust that the certificate actually belongs to your friend. The trust relationships between users of PGP form a trust hierarchy called a "web of trust".

Transport Layer Security (TLS) [8] is a security protocol that operates on the transport layer. It is used to secure the connection between two applications. This differs from IPsec, which secures a connection between two nodes. TLS is an updated version of the Secure Socket Layer (SSL) [31] protocol developed by Netscape. When a TLS connection is initiated the responder tells the initiator its public key by sending its public key certificate. The connection initiator can then authenticate the responder using the received certificate. The public key found in the certificate can be used to securely transfer a shared secret to the responder, and based on that secret the session keys can be generated. After that, the transmission of data between the two nodes is secured by encryption using the session keys.

2.3 Mobile telephone networks

The concept "mobile telephone networks" encompasses many technology layers and aspects. However, in this section only the aspects relevant for this thesis are explored. This includes, but is not limited to, network structures, communication and security. The focus will be on systems used in Europe, namely GSM [32], GPRS [1] and UMTS [2] for second generation (2G), "second and a half generation" (2.5G) and third generation (3G) systems respectively.

2.3.1 Second Generation

The Global System for Mobile Communications (GSM) is the most widely used 2G system today. According to [33] there are over 200 countries/areas that offer GSM related services (GSM, GPRS, EDGE, 3GSM). GSM was taken into operation in 1992

[34]. Since then the standard has been developed in phases, each phase introducing new features or otherwise enhancing the GSM standard. The General Packet Radio Service (GPRS) was introduced in Phase 2+ [35]. GPRS is often also called a 2.5G system. The reason for this is that GPRS offers 3G-like services on a slightly modified 2G network architecture.

With GPRS it is possible to use packet switched data services. This includes web surfing, using e-mail and file transfer. The fact that GPRS has a higher connection speed than GSM facilitates the new services. The theoretical maximum speed of GPRS is 171,2 kbps compared to 9,6 kbps for GSM. The GSM system architecture has not had to change very much to facilitate GPRS services. The difference between the original GSM architecture compared to the GPRS enhanced one is two added network elements: the GGSN and the Serving GPRS Support Node (SGSN). The GPRS system architecture is shown in Figure 9. The picture is based on information from [36] and [34].

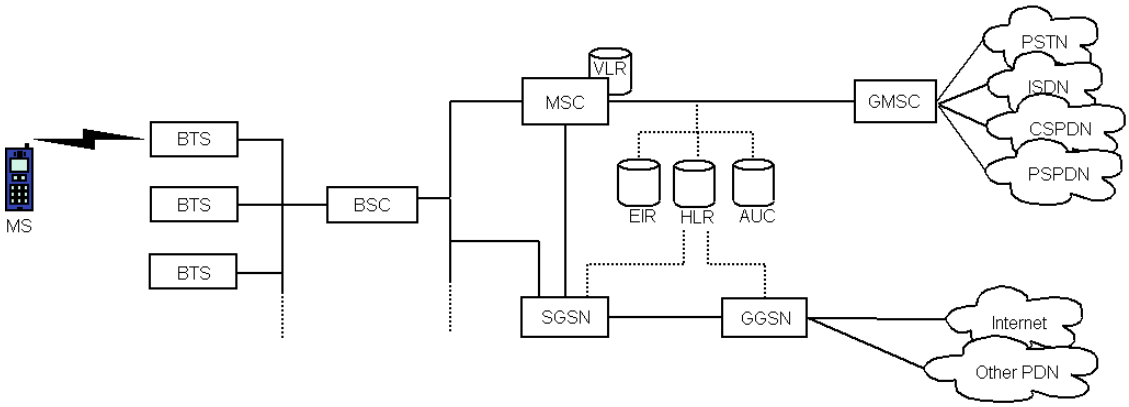

Figure 9: The GPRS system architecture

A GPRS network has six basic nodes. A Base Transceiver Station (BTS) covers a certain area around it. Mobile Stations (MS) that are within that area connect to the mobile telephone network via the BTS. A Base Station Controller (BSC) manages radio resources and handles handover between BTSs that are under its control. The Mobile Switching Centre (MSC) is a switching node that works much like a local exchange of a fixed network. Apart from switching, an MSC also handles functionality related to MS registration and authentication, handover and roaming. The Gateway

MSC (GMSC) is a specialized MSC that handles connections between the current network and other networks, e.g. another Public Land Mobile Network (PLMN) or the Public Switched Telephone Network (PSTN). All connections to and from the PLMN go through a GMSC. There can be multiple BTSs, BSCs, MSCs and GMSCs in a GSM or GPRS network.

In addition to the nodes mentioned earlier, a GSM network also has four types of databases. Each MS must be registered to a network and the Home Location Register (HLR) contains this registration information. The location of a MS is constantly updated into this database. The information is used to forward incoming calls to the network where the MS resides. A Visitor Location Register (VLR) is used to store information about MSs that are located in the service area of a MSC. That is, the area covered by the BTSs belonging to the BSCs that belong to the MSC. The information in the VLR is used to locate the MS when the network receives a call for the MS. The Authentication Centre (AUC) stores security information, e.g. encryption keys that are used to facilitate encrypted traffic. The Equipment Identity Register (EIR) stores mobile telephones identities. It is used to hinder the usage of barred mobile telephones.

The difference between the architectures of a GPRS and a plain GSM network, as stated earlier, are the two nodes: SGSN and GGSN. In addition the HLR has some enhancements to handle GPRS subscriber data and routing information and the BSC to handle packet data. The GGSN acts as the gateway between the PLMN and other packet data networks such as the Internet. It is the packet switched data equivalent to the GMSC. The task of the SGSN is to handle data delivery for the MSs belonging to its service are. The functionality is much like that of an MSC, but for packet switched data. It also takes care of connection establishment to the GPRS network. The GPRS network is IP based.

When a MS wishes to establish a connection to a packet data network (PDN), such as the Internet, it first sends an Activate PDP Context request message to a SGSN. The message includes Protocol Configuration Options such as host authentication and configuration options. The SGSN forwards the received message to a GGSN. The

GGSN processes the received message and then sends back an Activate PDP Context response message to the SGSN. Depending on the cause value received in the response message the SGSN sends either an Activate PDP Context Reject or Accept message back to the MS. If the context activation was successful the MS is free to begin connecting to nodes in the PDN using the IP-address assigned to it by the GGSN.

2.3.2 Third Generation

By enhancing the GSM system we got the GPRS system. The 3G system UMTS is the next step from GPRS. The other big 3G system besides UMTS is CDMA2000. Both systems are operational [37]. In this thesis the focus will be on UMTS. The big improvement introduced to GSM by GPRS was the higher data rates and the possibility of packet switched data. These are also the most important features of UMTS and 3G in general. With UMTS the data rate can be as high as 2Mbps [38]. With the capacity of UMTS there are many new services and ideas that can be realized, e.g. video telephony.

The structure of the UMTS system architecture is very much like the architecture of the GPRS system. The UMTS architecture is shown in Figure 10. If the UMTS architecture is compared with the GPRS architecture shown in Figure 9 one can see that there seems to be only a few differences. However, the main difference between the two networks (GPRS and UMTS) is not the network architecture but the used transmission methods and the used protocols. In UMTS the mobile telephone, or other connected apparatus, is called a User Equipment (UE) instead of a MS. Where the BTS used to be in the GPRS architecture are now nodes called Node B. Node Bs are also called base stations (BS). The functionality of a Node B is very similar to that of a BTS. The major differences between these two nodes are the transmission methods enabling the high data rates of UMTS. Also the BSC of the GPRS network is replaced by a new node. In UMTS the node is called the Radio Network Controller (RNC), which similarly to the BSC handles radio resources. Apart from these changes also the

MSCs and SGSNs have to be modified to support UMTS. Still, connection establishments and Context Activations work in a similar fashion as with GPRS.

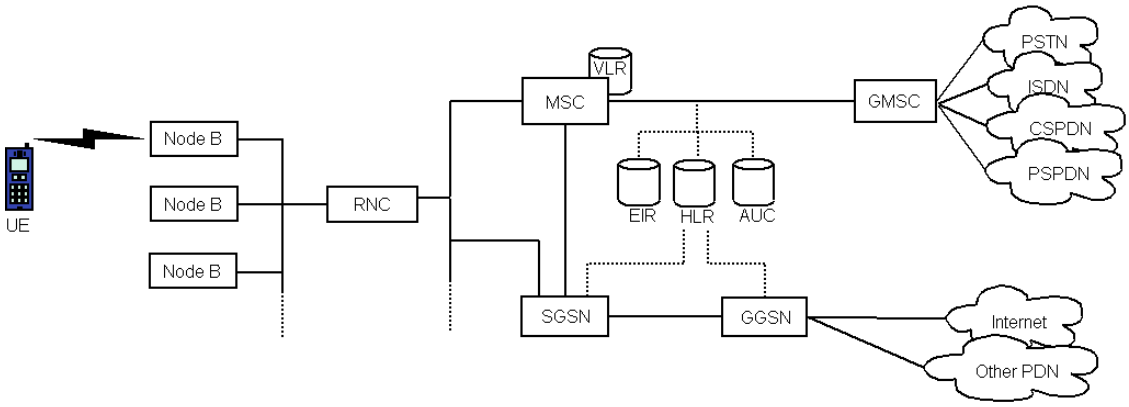

Figure 10: The UMTS system architecture

2.4 Security in mobile telephone networks

Just as computer networks need security so do also mobile telephone networks. There are four main concerns; unauthorized use of the network, unauthorized listening of communication, use of stolen MS and unauthorized access to subscribers identities [34]. That is to say that both the network and its users have to be secured. This section will focus on 3G security [39], beginning with a short report on some improvements that have been done to 2G security to reach the security level of 3G [40].

In GSM the confidentiality of data is ensured by using encryption over the air interface. However, in the rest of the network all information travels in plain text. This problem has been addressed in UMTS. There are security mechanisms for protecting information in and between networks. The security in GSM is provided in the base stations (BTS) while in the UMTS the security functions are moved to the switches, the MSC and the SGSN. This results in secure communication between the MS and the switch in UMTS compared to security between MS and BTS in GSM. Another improvement to 3G security is the increased key length compared to 2G. This allows for stronger security algorithms. This is important since the algorithms used by 2G

have proven to be too weak [41]. In GSM a MS has to authenticate itself to the network but the opposite is not true. This allows for different fake base station attacks. Many such attacks and other attacks are described in [39] along with information about how 3G possibly counteracts them.

2.4.1 Protection against unauthorized mobile equipment

The size of the mobile telephone and the price makes it an attractive piece of equipment for thieves. Stolen, or equipment that otherwise is not allowed to connect to the network can be barred from the network. The EIR database contains information about the identity of mobile equipment. When an UE connects to the network the MSC can request the international mobile equipment identity (IMEI) of the UE. To check that the IMEI is valid it can use the EIR database. If the IMEI is either barred or not present in EIR the connection is refused.

The telephone itself and the Subscriber Identity Module (SIM), or the User Services Identity Module (USIM) in 3G, are also protected against unlawful use. The user needs to be authenticated before he/she gains access to the UE. This is achieved by requiring that the user knows a secret, a personal identity number (PIN) code. The UE can also be configured to only allow authenticated USIMs to be loaded into it. In this case the USIM and the terminal in which it is going to be used need to share a secret.

2.4.2 Protecting the subscriber identity

The reason for protecting the identity of the subscriber is to protect the privacy of the subscriber. That is, the information about the movements of the subscriber and used services should not be revealed. This is achieved by using a temporary identity for the subscriber when possible. The true identity of the subscriber, the international mobile subscriber identity (IMSI), is only used when a temporary identity cannot be used. In general that only happens when the UE is connecting to the network and has not yet

been assigned a temporary identity. After the temporary identity has been assigned it is used instead. It is also important that the same temporary identity is not used too long because then the privacy of the temporary identity can be compromised. Another reason for not using the true subscriber identity is that if it would fall into the wrong hands it could be exploited.

2.4.3 Subscriber authentication

The base for subscriber authentication in UMTS is a 128-bit secret key only known by the USIM and the AUC database in the home network of the subscriber. When the UE registers itself at a network, the network first has to authenticate the UE. The SGSN (or MSC) queries the AUC database in the home network of the UE and receives authentication information. This information includes keys and values calculated by the AUC based on the secret key of the UE. The UE authentication is initiated by sending it a random value received from the AUC. The UE makes some calculations based on the random value and its secret key. The result is sent back to the SGSN. The value is compared to the expected response that also was received from the AUC. If it is a match the UE is authenticated. The network is authenticated to the UE based on sequence numbers embedded into the random value and other data sent to the UE from the SGSN. More about this can be read from [39].

2.4.4 Confidentiality of communication

Based on the random value sent to the UE during authentication, the UE can calculate session keys for confidentiality and integrity purposes. If the authentication is a success the keys are used to encrypt the traffic that follows. The SGSN receives the corresponding keys from the AUC with the data that is used for authentication. The confidentiality and integrity algorithms to be used in UMTS systems are defined by 3GPP [42] in [43] and [44].

3 Problem Statement

In this chapter, the motivation for a HIP proxy in a 3G system is given, and the difficulties related to the work are presented. Implementing HIP in a 3G system affects both the end-hosts and the network. These issues are discussed and problems are identified. A connection between a legacy UE and a HIP enabled host is inspected step by step, pointing out what needs to be done to the current architecture to facilitate the connection via the HIP proxy. Finally, the evaluation criteria are presented.

3.1 Advantages of using HIP in a 3G system

When using packet switched data services in a 3G network, the traffic usually travels outside the PLMN, i.e. in the Internet, for a part of the time. The Internet is a public network that by design is insecure. The authentication, confidentiality, and integrity features provided by HIP can be used to perform secure communication over the public network part of the connection.

The cryptographic keys used in HIP are coupled to the identity of the host; the identity is the public key of an asymmetric key pair. Consequently, if a host knows the identity of another host it can communicate securely with it. Conversely, since the identity can be confirmed with the cryptographic keys, hosts always know the identity of the HIP-communication peer(s). Stealing someone's identity requires also stealing the private key. Without the private key incoming messages cannot be decrypted, thus the stolen identity is useless without it. This makes the identities quite secure, with the security relying much on the key length and the privacy of the private key.

The Internet is constantly evolving; one of the current trends is improving the support for mobility and mobile nodes. Thus many of the current research projects focus on mobility, and HIP is no exception. One of the strong points of HIP is the implicit support for mobility. HIP mobility makes it possible for the connected peers to change

their position in the network topology, still being reachable by other nodes and maintaining existing connections. This is possible since the connections are actually established between two identities instead of two IP addresses. The separation of identifiers (HI) from locators (IP address) results in that the used locators do not affect the connection itself, but only the path the packets travel. When a node moves to a new topological position it sends location update messages to the connected peers informing them of its new position, i.e. the new locator.

The Internet is an insecure network where data mostly travels in plain text. Using HIP, the data is protected by encryption, the hosts are authenticated, and the information transmitted between the two communication parties stays confidential. Further, attacks that try to fake the identity of the peer are efficiently made unviable. These are all features that would be beneficial for 3G users connected to the Internet. Since the Internet is evolving and getting increasingly more mobile, it can be expected that web service providers will also become mobile. This means that in the future it might be quite common that servers are not always found behind the same IP address. With HIP this is not a problem; the IP address of the HIP enabled server is not the key issue when making, and maintaining, a connection to it. The identity of a host plays a major part in both security and mobility features of HIP.

Security and mobility are not the only improvements provided by HIP compared to traditional IP based traffic. One important feature is the support for multi-homing. Multi-homed hosts can effectively use multiple network interfaces in parallel. This feature can be very valuable if e.g. one of the interfaces is connected to a link that suddenly gets heavy traffic loads, or even dies. In this case, all traffic that was going through that interface can be routed to the destination host via some other active interface connected to the host.

Besides the identifier and locator separation, all the features provided by HIP can also be provided by a combination of other services such as IPsec and Mobile IP. However, by using HIP to gain these features only one solution is required, thus making it more desirable. HIP is a protocol that encompasses many of the currently identified needs, so

using HIP to gain these features will not result in any interoperability problems. The increased security over IP networks provided by HIP can, along with the idea of a host identity, be used to provide new services. Different services that require the user to be authenticated, e.g. e-commerce related services, can benefit from using HIP; a HIP connection is secured by means of cryptography and a peer is always aware of the identity of the other peer. Also the multi-homing feature of HIP can be benefited from when designing new services.

3.2 Integrating HIP as part of the 3G system

HIP is designed to be an end-to-end protocol. Thus the optimal solution for using HIP in a 3G system would be that all UEs connected to the network would understand HIP. In that case the problem of having HIP communication between nodes in the telephone network and nodes in the Internet would be simplified. It would equal to the regular HIP scenario where two nodes communicate using HIP; each UE would have a HI of their own, and when a UE wants to initiate a connection it would be assigned an IP address during context activation, just as in the current architecture. The HIT corresponding to the HI, and the assigned IP address could then be used in the same manner as a regular HIP node uses them. However, this scenario is not going to become reality in the near future. Each UE that is not equipped with a modified IP-stack during manufacturing would need to be updated. This means every UE that has been manufactured so far. In addition, with high probability all UEs that will be manufactured in the near future will not have the modified, HIP-enabled, IP-stack. Manufacturers will not begin to consider implementing HIP in their UEs until HIP becomes widely used and there is a demand for it. One can argue that maybe not all UEs need to be HIP enabled, only those UEs that will be using HIP services would need to be updated to support HIP. While this is true, how can it be known which users, and thus UEs, will require HIP. Also, some service providers might insist that their service can only be used together with HIP. This would deny the service from all non-HIP UEs.

Technically, updating UEs to support HIP can probably be done quite easily. However, the problem is not simply to create an update for the UEs and get the UEs updated. A bigger challenge would be to get information about the update request spread and the users convinced of the necessity of it. This problem is similar to the challenge of updating the used Internet protocol from version 4 to version 6. It would be desirable to perform the update on all nodes at the same time. Then the network would first work with the old protocols and in the next instance all nodes would have been updated to use the new protocols. But this is only a theoretical solution and it cannot be performed in practice.

Instead of having all the UEs be HIP enabled, a dedicated node in the mobile telephone network could handle all HIP related communication. However, requiring network operators to invest money in new hardware might not be received very well. Also, modifying a thoroughly standardized system architecture is probably not the right way to go. All packet switched data, which is the traffic that would be benefiting from HIP, always flows through the same set of nodes in a 3G network. Thus one of those nodes might be a prime target for an update to enable HIP in the network. By making one of the nodes in this path able to understand and operate using HIP, at least part of the communication channel could be made to benefit from HIP: communication from that node forward could use HIP to protect traffic. This is depicted in Figure 11.

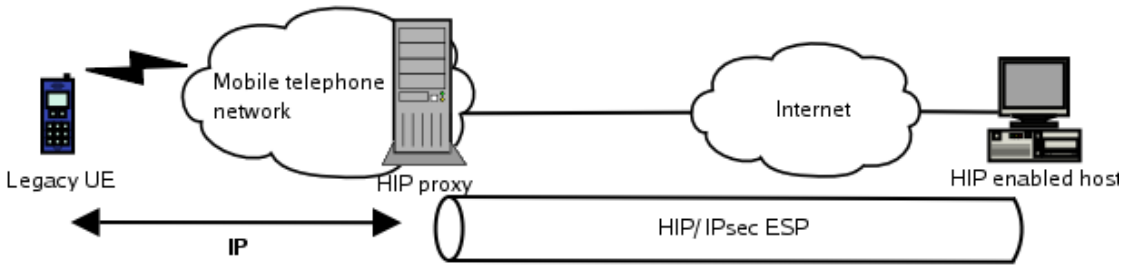

Figure 11: HIP proxy in a mobile telephone network

The connection between the proxy and the HIP enabled host could use HIP as any two HIP enabled hosts. However, the communication between the HIP proxy and the legacy UE would still just be “normal” network traffic; traffic that the network

supports e.g. IP. The task of the proxy would be to convert HIP traffic into normal network traffic, and normal network traffic into HIP traffic. Furthermore, the HIP proxy would be responsible for handling the HITs and SAs of the UEs towards the HIP hosts. Implementing the HIP proxy in a node outside the PLMN is not a reasonable solution. The result would be that HIP would be used between the HIP enabled host and the HIP proxy, but the traffic between the HIP proxy and the PLMN would use regular, unprotected, IP. In this case the features of HIP are not fully taken advantage of. Especially the security provided by HIP would be of no use.

There are some conclusions one can draw from the fact that some nodes will need modification to enable HIP in a 3G system. From a deployment perspective the best solution would be the one that affects the least amount of nodes in a network. The reason for this is that it would make deployment easier and less likely to fail or be performed only partially. When a UE uses packet switched services, the data always travels the same set of nodes: Node B, RNC, SGSN, GGSN. Depending on the actual network layout, the amount of the different nodes can vary. However, since network operators strive to have sensible and cost-effective networks, there are probably not as many SGSNs or GGSNs as there are Node Bs or RNCs. This would indicate that a SGSN or a GGSN would be the most appropriate node for a HIP update.

The GGSN serves as a gateway node between the UMTS network and external packet switched data networks such as the Internet. If the HIP proxy was to be implemented in the GGSN, then HIP services would be provided for the part of the communication that goes over the Internet. The rest of the communication channel, the part that traverses the mobile telephone network, would be secured by the network just as it is today. The interconnection point of two networks would be logical place to switch the used protocol.

In the current network the GGSN is responsible for allocating IP addresses to the connected UEs, which is another reason for locating the HIP proxy in the GGSN. The HIP proxy assigns a HIT to each of the nodes it serves, including all non-HIP UEs. The allocation is done because, in HIP, all packets are sent to a HIT instead of an IP

address. The HIP proxy uses a HIT to distinguish between incoming HIP packets destined at different UEs. Hence the UEs must each be assigned a unique HIT. Assigning an IP address and a HIT are similar tasks, therefore they could well be done by the same node. The GGSN could assign both IP addresses and HITs to UEs during context activation.

By no means is the GGSN the only possible node that could be used for running the HIP proxy, but when looking at its current tasks the functionality of a HIP proxy fits well in it. Also the network structure, with the GGSN on the border between the two networks, speaks for placing the HIP proxy in GGSNs. Based on this, the target node for which the HIP proxy will be designed in this thesis is the GGSN.

There are four possible connection scenarios for a HIP proxy in a GGSN:

1. A legacy UE connects to a legacy host
2. A HIP enabled UE connects to a legacy host
3. A legacy UE connects to a HIP enabled host
4. A HIP enabled UE connects to a HIP enabled host

Case 1. is straightforward, it is a normal communication between the UE and the host in the Internet just as it is done today. Case 2. is likewise rather uninteresting since it only allows for HIP protected communication inside the mobile telephone network, between the HIP UE and the HIP proxy. However, the mobile telephone network is not where the security features are needed. The 3G network has much improved security features compared to the 2G network, and the use of HIP there is not necessary. Cases 3. and 4. are the interesting ones. They make it possible to secure the connection over the Internet when the host in the Internet is HIP enabled. This thesis will mainly focus on Case 3., because presently there are no HIP enabled UEs and we want to bring the security provided by HIP into the network. However, the situation presented in Case 4. might be real some day, so the proxy needs to be able to handle it if/when it happens.

HIP has been designed so that the initiating host has to do heavier calculations before the connection can be set up. This may cause problems since the HIP proxy in the GGSN is the party that in the current architecture always initiates the HIP connection. It should be noted that in the future there may also be incoming connections, but currently connections are always established from the UE towards the host in the Internet. With many simultaneous connection setups the GGSN might get heavily burdened. This might result in long connection times, dropped connection requests or even in overloading of the GGSN.

3.3 Connecting a legacy UE and a HIP enabled host

It was concluded earlier that the UE will be a legacy host and it will not be aware of HIP, so there will be no changes to it or its functionality. The only node in the 3G network that needs modifications is the GGSN. In the future development phases some changes might also be needed in the HLR for providing enhanced HIP functionality, e.g. static HIs. The HIP connection will terminate in the gateway node (GGSN) between the PLMN and the Internet, and all traffic in the 3G network will remain unmodified. Outside the PLMN only small, non-3G related, changes are needed to enable the use of HIP. Basically all that is needed is that the destination node of the HIP connection is HIP enabled, i.e. has a modified IP-stack supporting HIP. To make the use of HIP more convenient, DNS should be modified to also return the HIT of a host along with the IP address(es). This is also needed for supporting mobile hosts. Alternatively, instead of the modified DNS, some other lookup service should be implemented. If only a small amount of HIP enabled servers are available, it might even suffice to manually configure HIT-IP mappings into the proxy. Based on the facts presented above, it can be seen that for getting a system up and running, the part that needs most attention is the HIP proxy implementation in the GGSN.

Before setting up a HIP connection on behalf of UEs, the GGSN/HIP proxy must assign HIs for the communicating UEs. The HIP proxy generates an asymmetric key-pair for each UE that requests a connection. From the public key a HIT is calculated for

the UE. The proxy must store this information for each legacy host so that it can perform cryptographic functions on the traffic of the UE using the correct key. When a UE requests a connection to a HIP enabled host, the HIP proxy should use the stored HIP information related to the UE to establish the HIP connection between itself and the HIP host. After the HIP connection is set up the communication can continue. When the UE sends packets to the HIP host, the HIP proxy catches the packets before they reach their destination. The proxy inspects the caught packet and finds the appropriate SA based on the identity of the UE and the destination host. Next, the proxy generates an IPsec ESP packet that contains the data packet sent by the UE. The ESP packet, which was encrypted by the proxy, is then sent to the HIP enabled host. Traffic from the HIP host towards the legacy UE must likewise be handled by the HIP proxy. It applies cryptographic functions on the received packets according to the corresponding SA. The correct SA is resolved using the SPI found in the ESP header of the packet. The true recipient of the packet is resolved from the destination HIT of the received ESP packet, and the decrypted data is sent to the correct UE.

The IP address of the HIP host, received from the DNS, may actually not point to the HIP host, but to a Forwarding Agent (FA) serving the HIP host. In this case the FA forwards packets received on a HIP association based on the destination HIT found in the packet. This works well as long as the HIP proxy only has to establish connections to a single host (host A) served by the FA; the proxy uses the resolved HIP information of the host (IP_{FA}, HIT_A) to establish the connection(s). When a legacy host connects to a host served by the FA, it initiates the connection with a packet with the IP address of the FA (IP_{FA}) as destination. When the HIP proxy receives the packet it uses the IP-HIT mappings to find the HIT corresponding to the IP address. However, a FA can serve multiple hosts. If the proxy needs to establish a connection to another host (host B) served by the same FA there will be a conflict in the HIP proxy; the IP-HIT mapping table will contain multiple HITs (HIT_A, HIT_B, etc.) for a given IP address (IP_{FA}). Now the proxy has no way of knowing which HIT corresponds to the desired destination host. In this case the HIT that will be associated with the IP address of the FA is implementation dependant – it might e.g. be the HIT of the host whose information was fetched first from DNS, but then again it might follow some other

logic. Regardless of method, the probability of a correct selection is about the same as if randomly picking a HIT; there is not enough information for concluding the correct HIT.

The legacy host is not aware of HIP and thus cannot directly utilize its features, such as the separation of identifier from locator. This leads to the described problem, where the legacy host uses the IP address of the FA for uniquely trying to identify a specific host. The HIP proxy needs to be informed of the identity of the peer, i.e. the HI or HIT, to be able to uniquely identify the connection peer. Assuming that the legacy host resides in a private network, and because an IPv6 address has the same structure as a HIT, the solution is quite straightforward. When the legacy host requests information about a HIP enabled host, instead of the HIP proxy replying with the IP address of the host, the reply could contain the HIT of the HIP host. The legacy host would not know the difference between an IPv6 address and a HIT, so it could use the HIT as destination address for its packets. The private network routes the packet to the proxy as if it was equipped with a regular IP address. When the HIP proxy receives an IP packet from the legacy host it knows that the destination of the packet might be a HIT. By checking the 2 most significant bits of the destination address it can conclude if the destination is a HIT or an IP address; a HIT always has a 2-bit prefix of 10 or 01 [11]. If the destination is a HIT, it can be used for setting up a HIP association, as long as the proxy knows the IP address of the peer. Otherwise, if the destination is not the HIT, but the IP address of the peer, the proxy replaces the IP address with the corresponding HIT, if any, and tries to establish a HIP association with the peer.

3.3.1 Obtaining destination HITs

The resolving process for HIs in a HIP enabled environment is similar to that of IP addresses in the current architecture, e.g. with the help of DNS or other similar lookup services. The response from a DNS contains the HI and/or HIT along with the IP address(es). When a UE requests information about a host in the Internet, the DNS query travels through the mobile telephone network in the same way as packet switched data. The GGSN forwards the query to a DNS and receives a response from

the DNS containing the IP and HIT of the node if the requested host is HIP capable. The GGSN examines the response and saves the host information, since it will need both the HIT and the IP address when a HIP connection will be set up. The function of the GGSN is similar to maintaining a DNS cache since it still has to store DNS entries. Adding support for HIP to the DNS is not a trivial task and it is currently being researched in the HIP working group at the IETF. Information regarding HIP with DNS can be found in [45].

The use of a HIP enhanced DNS will not be available during the work on this thesis. The prototype of the HIP proxy will not be obtaining the HIT-IP pairs from a DNS but a tool/mechanism will be constructed for configuring HIT-IP pairs manually into the proxy. For incoming packets the HIP proxy will consult a table containing the pre-configured HIT-IP mappings. Based on the destination HIT the corresponding IP address is retrieved from the table, or vice versa. This method could well be used in small-scale networks with only a few hosts that need to have their information available. However, for bigger networks, e.g. the Internet with thousands of hosts, this solution is neither attractive nor feasible. A working solution for big networks could be to maintain an up-to-date list of the HIP hosts, but it would require an automatic update mechanism. The UE might also request a connection to a HIP enabled host that is not pre-configured in the list, which also is a problem with this approach.

3.4 Evaluation criteria

What are the important characteristics of a good HIP proxy? It would be desirable that the HIP proxy would be secure and protect against unauthorized usage and possible attacks. Users must be able to trust the devices and methods that provide security for their traffic. Security of a system is only as strong as the weakest link. The HIP proxy should not be that weakest link. One of the more important parts, security wise, is the HIP proxy configuration tool used for configuring HIT-IP mappings in the proxy. Only authorized persons should be allowed to modify the HIP proxy configuration.

Another important aspect of the HIP proxy is its performance. Since it is situated in a GGSN it may have to handle multiple connection establishments simultaneously. The HIP proxy should be able to perform well under heavy traffic. Also robustness is important, the HIP proxy should be able to recover from unexpected results. Other important characteristics include fault tolerance, stability and low complexity. The proxy should be able to handle possible errors. The design should also be simple enough. A too complex design might induce unnecessary errors.

Many of the desirable attributes of a HIP proxy that have been mentioned are quite abstract. There is no easy way of measuring them, e.g. it is difficult to prove that something is secure. By testing one can only prove the presence of faults but not that there are no faults or that the system is secure. The evaluation criteria are listed in Table 1 below.

Table 1: Evaluation criteria

- HIP proxy security
- Configuration tool security
- Performance
- Robustness
- Fault tolerance
- Stability
- Low complexity

4 HIP with a HIP proxy

The chapter begins with a presentation of HIP and the use of its specific attributes, such as the Host Identifiers (HI) and the Host Identity Tags (HIT). Different HIP packet structures, and the connection establishment called the HIP base exchange, are examined. Also regular HIP traffic will be described. Once the basic operation of HIP has been presented we take a look at the HIP proxy and its functionality. First a general description of a HIP proxy is given; how a HIP proxy handles a connection between a legacy host and a HIP host. Finally we inspect the case where the proxy is located in a GGSN, operating between the Internet and a mobile telephone network.

4.1 Host Identity Protocol

The central aspect for all the features of the Host Identity Protocol (HIP) is the identity of a host. Each HIP enabled host has at least one Host Identifier (HI) that is used to identify the node. In HIP, the HI is defined to be the public key of an asymmetric key-pair and it can be used for cryptographic functions as described in Section 2.1.3. In the following sections the actual use of HIs, HITs, and HIP in general will be described. It is assumed that IPv6 is the Internet Protocol used at the network layer.

4.1.1 The HIP base exchange

In HIP, communication between two hosts is always protected with IPsec. The HIP base exchange, a four-way handshake, is used to negotiate IPsec SAs between the communicating nodes. In HIP, the host that wants to initiate the communication is called the Initiator and the peer is called the Responder. Before the Initiator can begin the connection establishment with the Responder, it needs to find out certain information about the Responder. The required information includes the IP address, giving the current location of the peer node, and preferably a HIT. The Initiator can get

hold of these either from a DNS or from some other lookup service. If the DNS is used, it needs some modifications so that it can maintain information of the HIT(s) of a host and include it in the response. In some cases it is possible that the Initiator already knows the HIT of the Responder. After the Initiator has received the IP address and the HIT of the peer, it can begin the HIP base exchange as depicted in Figure 12 [46]

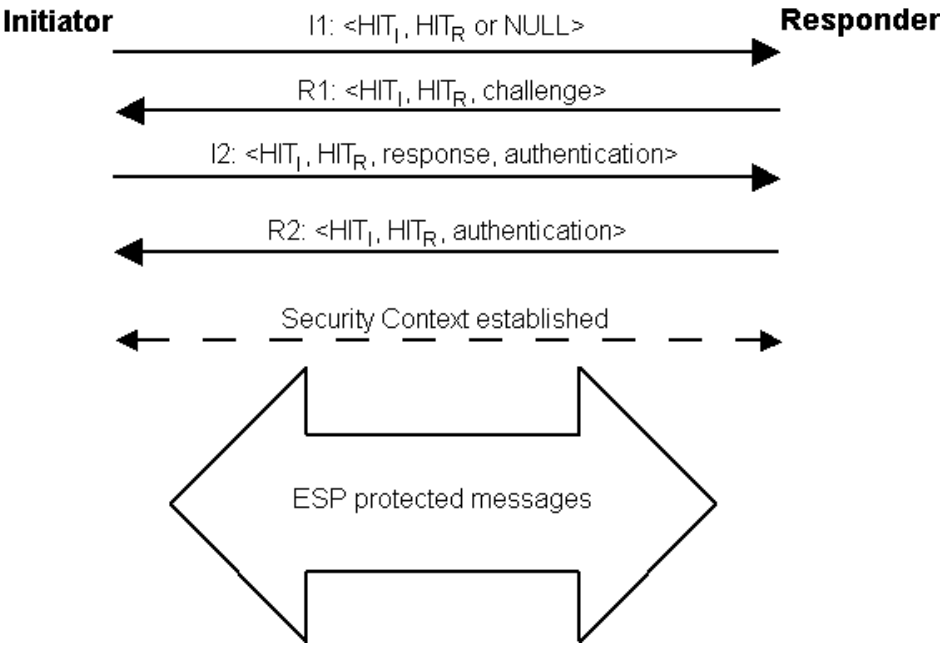

Figure 12: The HIP base exchange

The following description is based on the current version of the draft, ”draft-moskowitz-hip-09.txt”. It is, however, expected that some message parameters, and even complete messages, will change while the work on this Internet-Draft continues at the IETF. The following subsections describe the HIP base exchange on a per packet basis.

4.1.1.1 Packet I1

The base exchange begins with the message I1 sent by the Initiator. Basically the I1 packet is just a trigger for starting a HIP negotiation. It contains the HIT of the Initiator (HIT_I) and possibly the HIT of the Responder (HIT_R). If the Initiator did not find out

the HIT of the Responder via DNS lookup or other used lookup service, HIT_R can be set to NULL. This is known as opportunistic mode. The Responder can be configured to not accept opportunistic mode connection attempts.

Even if opportunistic mode HIP is allowed, such a connection attempt might not be successful e.g. if the obtained peer IP address points to a Forwarding Agent (FA) instead of the peer node. Without the HIT the FA does not know where to forward the I1 packet.

In the case that the I1 packet is sent to a legacy host the reply might be an ICMP destination protocol unreachable message. This tells the Initiator that the peer does not understand HIP. Alternatively the legacy host might drop the I1 packet.

4.1.1.2 Packet R1

The R1 packet is sent as a response to a received I1 packet and it starts the Diffie-Hellman procedure. The R1 packet contains both HIT_I and HIT_R. If the Initiator used opportunistic mode the responder can freely choose a HIT for any of its own HIs, otherwise the received HITs must be used. In addition to the HIP base header, the packet also contains:

- BIRTHDAY_COOKIE: This parameter is a combination of two functions that were considered to belong together. The birthday value is a reboot counter that is increased when an SA timeouts or when a host reboots. It is used for protection against replayed packets pretending reboot or loss of state of the peer. Cookie is a challenge that the Initiator must solve before the connection establishment procedure can continue. The Initiator proves that it is sincere by solving the puzzle and using some computing time for establishing a connection.

- DIFFIE_HELLMAN: This parameter contains the Diffie-Hellman key of the Responder. Together with the Diffie-Hellman key of the Initiator, it is used to generate an IPsec ESP session key.
- HIP_TRANSFORM: The parameter contains, in order of preference, the supported integrity and encryption algorithms supported by the Responder.
- ESP_TRANSFORM: In order of preference, the ESP modes supported by the Responder.
- HOST_ID: The Host Identity of the Responder.
- HIP_SIGNATURE_2: A digital signature calculated over the whole HIP packet with HIT_I and the checksum set to all zeros.

The use of the challenge makes connection initiation expensive for the Initiator since it requires the Initiator to perform heavy calculations to solve it. On the other hand the result verification done by the Responder at a later stage is a light task. More detailed information about the challenge can be found in [11]. Using the cookie mechanism makes Denial of Service (DoS) attacks expensive for an attacker.

The Responder can send challenges of different difficulties based on the amount of trust it has for the peer. Furthermore, the Responder can have in advance prepared R1 packets that it can send to the Initiator. This is possible since the signature over the packet is calculated with HIT_I and the checksum set to all zeros. Thus the content of the packet is known in advance and the signature can be calculated in advance.

HIP defines IPsec usage as mandatory and the needed keying material is generated using the Diffie-Hellman session key. Based on the received information in the parameters HIP_TRANSFORM and ESP_TRANSFORM, and the preferences of the Initiator, an incoming IPsec ESP SA is established. Keys for the SAs and the HIP packets are extracted from the keying material.

4.1.1.3 Packet I2

After solving the puzzle received in the R1 packet, the Initiator responds with an I2 packet. Both of the HITs are again present in the HIP header and must equal the HITs used previously. The I2 packet contains the following parameters:

- SPI_LSI: This parameter contains the SPI value that the Responder must use when sending IPsec protected data packets to the Initiator. For IPv4 compatibility purposes, the LSI value is included. The Responder should use this LSI value to represent itself when sending packets to the Initiator.
- BIRTHDAY_COOKIE: The cookie in this packet contains the solution to the challenge received in packet R1. The birthday has the same role as in packet R1.
- DIFFIE_HELLMAN: This parameter contains the Diffie-Hellman key of the Initiator. Together with the Diffie-Hellman key of the Responder, it is used to generate the IPsec ESP session key.
- HIP_TRANSFORM: The parameter contains the integrity and encryption algorithms selected by the Initiator based on information received in packet R1.
- ESP_TRANSFORM: Contains the ESP mode selected by the Responder based on information received in packet R1.
- HOST_ID: The Host Identity of the Initiator. It is encrypted using the selected algorithms and the keys generated by the Initiator after receiving the R1 packet.
- HIP_SIGNATURE: A digital signature calculated over the whole HIP packet.

For IPv4 compatibility purposes, an LSI is defined. The HIT cannot be given to IPv4 applications, so a 32-bit LSI is used instead. Because of the length of the LSI, collisions are possible. An LSI must always be unique between two communicating hosts, so a host may need multiple LSIs to represent itself towards different peer nodes. Upon receiving the packet, the host must check that the proposed LSI matches the HI.

Before processing the received I2 packet, the Responder may confirm that the packet is a response from a host to which it has recently sent an R1 packet. The Responder verifies the puzzle solution in the cookie parameter to make sure that the Initiator has solved it. If it was solved correctly, the Responder uses the received Diffie-Hellman key to generate keying material for itself. The HI of the Initiator can be decrypted using a HIP key extracted from the keying material. The SAs of the Responder can be created using the keys derived from the keying material and the information received in the parameters HIP_TRANSFORM and ESP_TRANSFORM.

4.1.1.4 Packet R2

The R2 packet finalizes the HIP base exchange. Like with the previous packets, the correctness of the included HITs must be checked. The main task of this packet is to help the Initiator to establish the outgoing SA and to authenticate the peer. The R2 packet contains the following parameters:

- SPI_LSI: Similar functionality as when sent in the I2 packet.
- HMAC: A HMAC [47] hash calculated over the packet.
- HIP_SIGNATURE: A digital signature calculated over the whole HIP packet.

To authenticate the sender of the R2 packet, the Initiator must verify the HMAC and the signature. If all verifications of the R2 packet are successful the HIP connection between the Initiator and the Responder is established.

4.1.2 HIP traffic

During the HIP base exchange an IPsec SA pair is created between the hosts. The negotiated SAs are bound to HITs instead of the IP addresses as in normal IPsec. Because IP addresses still are used as locators for the nodes, the use of HITs with the SAs would indicate that the regular packet structure cannot be used as such but the

HITs need to be included somewhere. However, the actual packet structure stays unchanged, the changes are found in the logical packet structure. It has previously been stated that in a HIP modified IP-stack an IP address to HIT translation is performed at the HI layer. An incoming ESP protected packet can be linked to the source HIT using the SPI; the incoming SA can be found based on the SPI, and the HIT can be found from the SA. Logically, a HIP connection between two HITs is used, but between the Internetworking layers of the two nodes, the connection is in fact a regular ESP protected IPsec connection. The difference between the logical and the actual packet structure is depicted in Figure 13 [25].

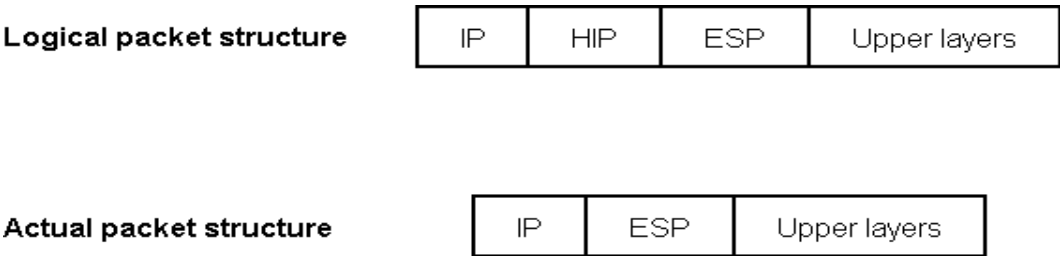

Figure 13: The logical and the actual packet structure of HIP

4.1.3 HIP packet structure

An earlier statement was that data traffic over a connection that uses HIP does not actually contain a HIP header. However, HIP specific packets, such as the packets belonging to the HIP base exchange, do use a HIP header. The HIP header is an IPv6 extension header. The structure of the HIP header is shown in Figure 14 [11].

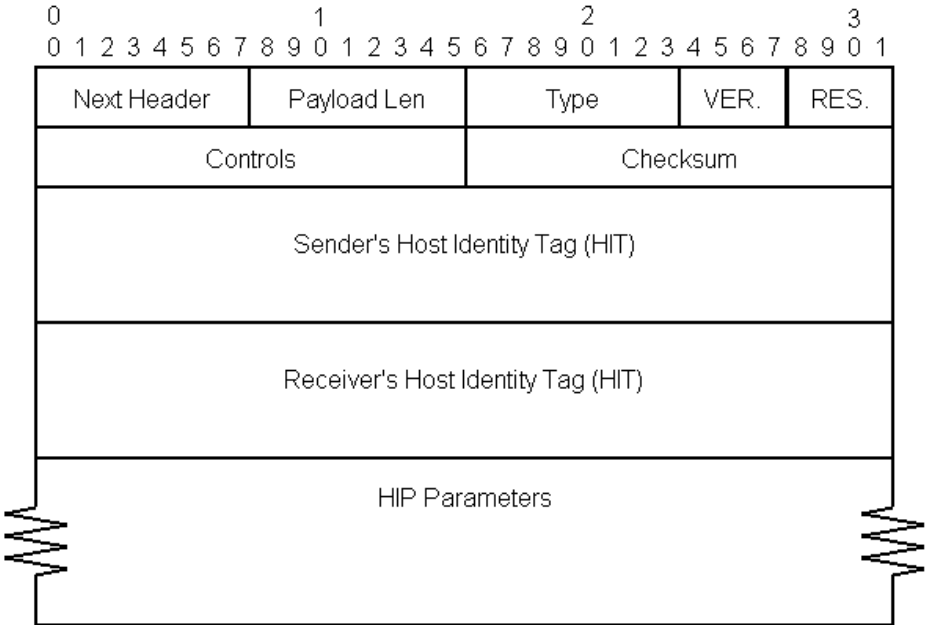

Figure 14: The HIP header format.

In the HIP header, the Next Header field indicates the next header type following this one. Payload Length gives, in 8 byte blocks, the length of the header excluding the first 8 bytes. The Type field gives the packet type, e.g. I1 or R2. The version field specifies the HIP version and the 4 bits following that are reserved for future use. The next two bytes are used for different controlling purposes. Three controls have been defined so far:

- C – Certificate: there are certificate packets following this packet.
- R – Resync Indication: the sender has lost its state and is trying to re-establish the connection.
- A – Anonymous: the HI of the sender is anonymous and should remain that way. The received HI should not be stored.

The Controls field is followed by the header checksum, and the HITs of the sender and the receiver as well as the HIP parameters. The HIP parameters are presented in a Type-Length-Value (TLV) format and include mainly the parameters presented in Section 4.1.1 regarding the HIP base exchange.

4.1.4 Additional HIP packets

The HIP base exchange described in Sections 4.1.1 uses four HIP specific packets. Apart from those four packets only four other HIP specific packets have so far been defined; one for bootstrapping when trying to connect to a host whose information is not known, one for sending certificates, one for sending unencrypted data and one for rekeying. Regular HIP communication uses normal IPsec ESP packet structure. More HIP specific packets will be defined during the development of HIP when new functionality is designed.

A host may want to create new SAs to be used instead of the old ones when e.g. the local policy requires it, or if the ESP sequence number is getting too big. The packet used for initiating this rekeying process is called the Update packet. Since SAs are bi-directional the host receiving an Update packet must send back a response to finalize the creation of new SAs in both directions. The packet contains a new SPI value to be used with the new SA as well as an index that gives the location from where the new keys should be extracted from the keying material. If the keying material has been exhausted, the packet can contain a new Diffie-Hellman key to be used for generating new keying material. The packet is protected using a HMAC and a digital signature. An Update packet always contains both the old and the new SPI, enabling intermediate systems, e.g. nodes performing Network Address Translation (NAT) [48], to update the SPI to IP address mappings. After a successful exchange of update messages, the communication should continue using the newly created SAs. The Update packet is also used for informing communication peers of a location change, giving the peers the new locator of the host.

The three remaining HIP specific packets are not fully specified yet and the processing of them is not defined. The broadcast packet, the HIP Bootstrap Packet (BOS), is designed for a situation where the Initiator does not know the IP address and HIT of the Responder. This could happen e.g. in a private network that does not have a DNS. In that case, a BOS packet is broadcasted to the local network to learn the information of the Responder. The HIP Certificate Packet (CER) can be used to provide more trust in

the Initiator, assuming that the Responder trusts the certification issuer. The CER packet is used for sending a certificate for the HI of the Initiator. Finally, the HIP Payload Packet can be used for sending non-ESP protected data.

4.2 The HIP proxy

HIP is designed to be an end-to-end protocol and as such, requires that both end hosts implement HIP. However, there will be situations when a legacy host and a HIP enabled host wish to communicate with each other. The legacy host does not understand HIP and is not aware of HITs and HIs. On the other hand, the HIP enabled host would prefer to take advantage of the services provided by HIP. Both parties can be pleased by adding a HIP proxy in the communication path. The proxy terminates the HIP connection on behalf of the legacy host.

Since security is the biggest advantage of using HIP instead of pure IP, the placement of the proxy cannot be arbitrary. This can be illustrated with an example: If both the legacy host and the HIP enabled host reside in the same network, a HIP proxy does not help much security wise; the same data goes both in plain and encrypted format in the same network.

The part of the connection that lays between the HIP enabled host and the HIP proxy uses the security functionality provided by HIP. However, the rest of the connection, from the HIP proxy to the legacy host, uses whatever security features are provided by the protocol used in that network. Usually, this means no security. An attacker understands to attack the weakest link of a connection; in this case the weakest point is between the proxy and the legacy host.

For the HIP proxy to be useful, the network in which the traffic between the legacy host and the HIP proxy travels needs to secure the traffic in some other way. Alternatively some additional security method, other than HIP, needs to be used between the legacy host and the HIP proxy. Based on this, the most logical placement

for the HIP proxy is between a secure network where the legacy host resides, and the public network (e.g. the Internet) where the HIP host is located. The legacy host is located in a private network that is connected to the Internet via some designated, and secured, points. The private network is assumed to be secure. If the border gateways of the private network have a HIP proxy implementation the optimal solution is reached. This is the setup that the following sections will be based upon.

4.2.1 Basic functionality of a HIP proxy

The main task of a HIP proxy is to allow a legacy host and a HIP enabled host to communicate with each other; the HIP enabled host using HIP, and the legacy host using some other protocol, e.g. pure IP. The operation of a HIP proxy can be simplified as functioning as the IPsec ESP termination point on behalf of legacy hosts. The actual system, however, is much more complex than only a tunnel endpoint. For each connected legacy host an SA pair needs to be created between the HIP proxy and each of the peer HIP hosts. The HIP proxy must be able to distinguish between packets destined for different legacy hosts, as it is possible that multiple legacy hosts want to communicate with the same HIP host.

If the HIP proxy receives a connection attempt from a legacy host to a HIP enabled host, the proxy initiates the HIP base exchange with the destination host. After the IPsec SAs have been established, the proxy can begin forwarding packets between the communicating peers. If the received connection attempt is towards a legacy host, the proxy cannot do anything else than forward the packet. So far we have only talked about legacy hosts initiating connections to HIP enabled hosts, but it is also possible that HIP hosts initiate connections. Furthermore, the HIP host initiating the connection can be located in the public network, but it can just as well be in the private network with the legacy hosts. This has to be considered when designing the proxy.

Three basic scenarios are identified for a HIP proxy in action. The connecting end-hosts may both be HIP enabled host or both be legacy hosts without HIP capabilities.

The third alternative is that one of them is a HIP enabled host while the other one is a legacy host. This is also the most interesting case since it actually requires a HIP proxy between the hosts to enable HIP between them.

A legacy host initiating a connection to a HIP enabled host is the primary HIP proxy scenario. Before continuing with forwarding the packets from the legacy host the proxy should perform the HIP base exchange with the HIP enabled host. If it is a HIP enabled host that initiates a connection to a legacy host, the HIP proxy should establish a HIP connection between itself and the HIP host. After that is done it would allow the HIP enabled host to communicate with the legacy host.

If both hosts are HIP enabled there is no real need for a HIP proxy in-between. However, there might still be a HIP proxy on the path between the two hosts. If that is the case, the optimal solution would be to detect this in the HIP proxy and let the hosts communicate without interfering. This would result in a HIP connection between the two end hosts and hardly any burden on the HIP proxy.

The last case is that both hosts are legacy hosts. In this case, just as the case with two HIP enabled hosts, the HIP proxy should not interfere with the communication. Since neither of the hosts understands HIP the proxy cannot do much else than function as a router on the communication path, possibly performing NAT.

4.2.2 Different scenarios for using a HIP proxy

The previous section described the basic tasks of a HIP proxy. In the actual solution there are various things that must be taken care of. The following subsections present the eight (two nodes, two node types, two possible connection directions) possible connection scenarios in detail, two scenarios per section. The descriptions are based on a network setup similar to the one depicted in Figure 15. Also, if not stated otherwise, it is assumed that the HIT of a host can be acquired from a DNS.

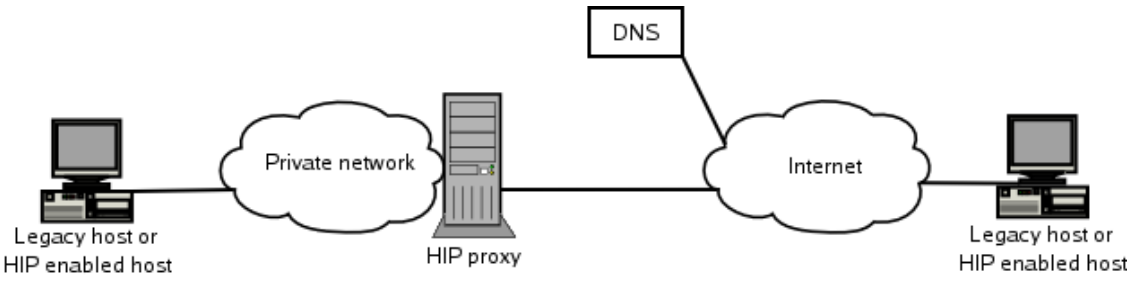

Figure 15: Network setup for use of HIP proxy.

4.2.2.1 Legacy host in private network – HIP enabled host in public network

The case where the legacy host is situated in the private network is the most interesting setup since it is the primary scenario when using a HIP proxy. HIP can be used to protect the traffic over the insecure network, the Internet.

The legacy host wishing to connect to a host outside of the private network first needs to acquire the IP address of the peer host. It sends out a DNS query to resolve the IP address. The query message passes through the HIP proxy to a DNS in the Internet. The response from the DNS contains the IP address of the peer host, and if the peer host is HIP enabled, also the HIT of the peer. The HIP proxy extracts this information from the response message and passes the response to the legacy host. The HIP proxy stores the extracted information for using it for establishing a HIP connection between itself and the HIP enabled host later, when the legacy host initiates the connection. If the legacy host is using IPv4, the HIP proxy needs to modify the DNS query to receive the HIT of the peer because the HIT is stored an AAAA record in DNS and returned only in response to an IPv6 address query.

If all HIP connections between the HIP proxy and the hosts in the public network, e.g. the Internet, were bound to the HIT of the proxy, the proxy would become very complex. The HIP proxy would not be able to separate incoming packets destined at different legacy hosts, especially if they were sent from the same HIP enabled host. The problem can be solved by letting the proxy generate an own identity (HI and secret

key) for each legacy host. Now the proxy can use the generated identity on behalf of the legacy host. With the connections bound to unique HITs, and thus also using different SAs, the HIP proxy can route the incoming packets to the correct nodes. This setup is depicted in Figure 16.

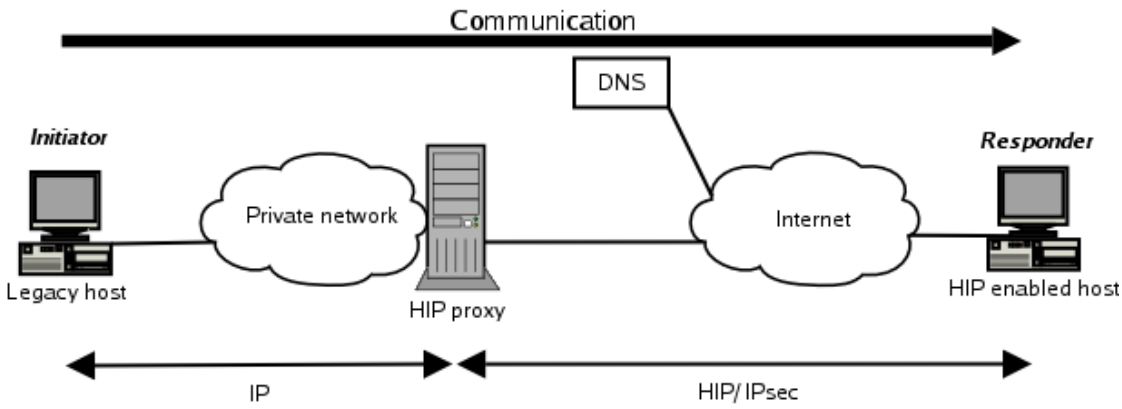

Figure 16: Legacy host behind HIP proxy initiates connection to HIP enabled host

The HIP proxy can also operate without the DNS mechanism if the destination addresses and HITs are configured into the proxy e.g. with a configuration tool. In this case the proxy does not need to read the DNS query response messages. When the legacy host initiates a connection, the HIP proxy checks the configured IP-HIT mappings to see if there is one for the IP address used as destination address by the legacy host. If no mapping is found, the peer host is assumed to not be HIP enabled host and no HIP connection is established. This system with pre-configured mappings is only feasible if operating in a limited environment where the HIP hosts are known beforehand.

So far, only the case where the legacy host initiates the connection has been discussed. The situation changes somewhat when the HIP enabled host is the initiating party. The HIP connection is now required between the connection initiator, the HIP enabled host, and the HIP proxy. Since the peer is not HIP enabled, the initiator can only get an IP address from the DNS. This leaves the host with two possible ways to initiate the connection. The host can either assume that the peer is not HIP enabled and use pure IP when initiating the connection, or it can try to use opportunistic mode HIP. One can

also consider the possibility that the IP address of the proxy and the HIT assigned to the legacy host by the proxy are configured into the DNS as information of the legacy host. In this case the HIP enabled host in the Internet will receive this information and initiate the connection with an I1 packet. This will result in a HIP connection being set up between the HIP host and the HIP proxy. The proxy will be able to forward the received packets to the correct legacy host based on the used HIT.

However, if the assigned HIT is not configured into the DNS and the initiator chooses to use pure IP, e.g. by sending out a TCP SYN message [49], the HIP proxy has limited possibilities to try to establish a HIP connection with the initiator. The HIP proxy could try to send out an I1 packet to each host trying to set up a non-HIP connection. However, this would generate a lot of excess traffic and is not a good solution in general. The conclusion is that if the connection is initiated with a non-HIP packet the connection will not use HIP.

Alternatively, if the Initiator wants to use opportunistic mode HIP, it sends out an I1 packet where the HIT of the Responder is set to NULL. If the peer would be HIP enabled the HIP proxy should let the packet pass through it to the peer. However, the HIP proxy might not know if the host in the private network is HIP enabled or not. In this case the proxy should hope that the peer is HIP enabled and pass the packet to the peer since not allowing two HIP enabled hosts to use HIP with each other would be a suboptimal solution. However, since the peer is a legacy host a HIP connection will not be established. An alternative scenario would be that the packet is sent out with the IP address of the proxy as destination. However, this does not work; when opportunistic mode is used HIT_R is set to NULL, so the proxy cannot know to which legacy host the packet actually is destined to.

To support opportunistic mode as described, there are different ways that the HIP proxy might acquire the information about if the host in the private network is HIP enabled. The HIP proxy might keep record of which hosts in the private network are HIP enabled. Alternatively it could also send an I1 packet to the host it wants to check, i.e. the destination of the incoming opportunistic mode I1 packet. If the HIP proxy gets

an R1 packet as response it knows that the host is HIP enabled. This is again a bad solution because of the excess traffic. If used, and if the received packet is an ICMP destination protocol unreachable message, or no response at all (even after some retransmissions), the host is a legacy host. In this case the HIP proxy should try to create a HIP connection between itself and the Initiator. The HIP connection protects messages for the legacy host while they are sent through the Internet. The two possible connection set-ups for this case are shown in Figure 17.

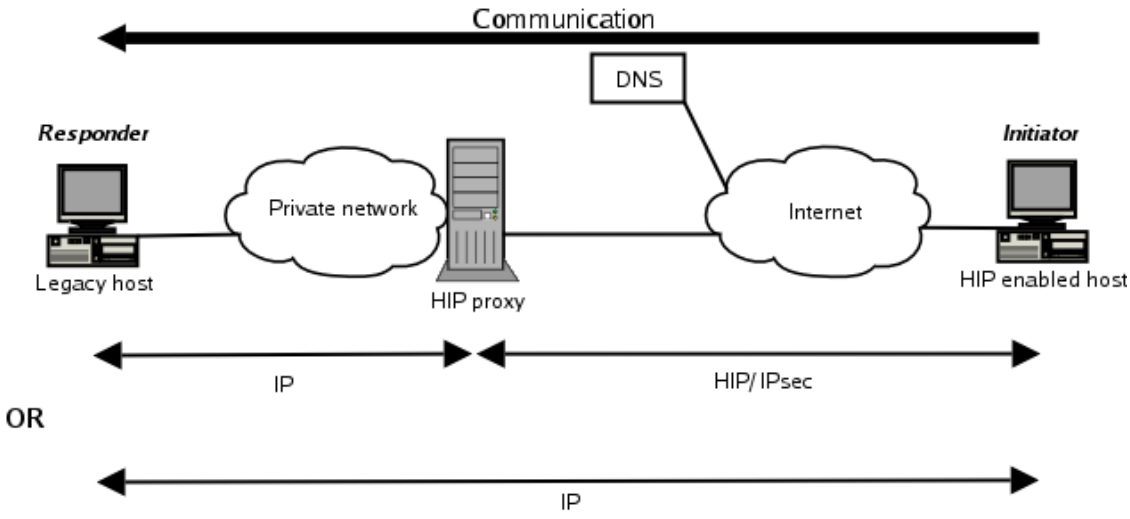

Figure 17: HIP enabled host initiates connection to legacy host behind HIP proxy

4.2.2.2 HIP enabled host in private network – Legacy host in public network

A reverse case compared to Section 4.2.2.1 is that the HIP enabled host resides in the private network. In this set up there is not much use for a HIP proxy, the traffic in the Internet will be unprotected since it is not possible to use HIP for protection. Naturally, it is possible to use some other method to secure the traffic over the Internet, but that is out of the scope of this thesis. The only part of a communication channel between the HIP enabled host and the legacy host that can be protected by HIP resides in the private network which is assumed to already be secure. Consequently there is no need to use HIP to protect traffic in the private network. The only benefit that could be achieved

with HIP in this setup is the mobility of the HIP enabled host in the private network. This requires that a HIP connection is set up between the HIP proxy and the HIP enabled node.

The legacy host initiating the connection sends e.g. a TCP SYN packet. The HIP proxy will let the packet through, possibly performing NAT, and the connection will be established between the legacy host and the HIP enabled host. HIP will not be used anywhere in the connection. If it is the HIP enabled host that initiates the connection it will first do a DNS query to acquire the IP address of the legacy host. The HIP proxy will again check the DNS response and notice that the peer is not HIP enabled since a HIT was not returned from DNS. Alternatively, if a configuration tool is used for defining HIT-IP mappings there will not be a HIT-IP mapping for the IP address of the peer. The HIP enabled host might still try to perform opportunistic mode HIP negotiation, but it will fail. If the HIP enabled host uses non-HIP connection establishment methods, the HIP proxy will let the packets flow through without modification. However, if the private network uses a private address space the proxy might perform NAT. This scenario is shown in Figure 18, the connection initiation can come from either direction.

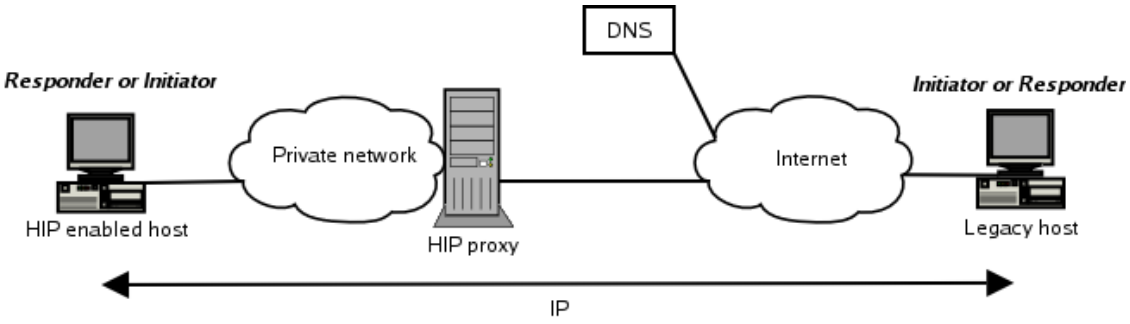

Figure 18: Connection between HIP host behind HIP proxy and legacy host

4.2.2.3 HIP enabled host in private network – HIP enabled host in public network

When there are two HIP enabled hosts communicating with each other there is no need for a HIP proxy in-between. A private network that has a HIP proxy on its border

might still contain both legacy hosts and HIP enabled hosts, and the HIP proxy needs to be able to handle both HIP and non-HIP connections.

Before the host in the private network initiates the connection, it queries the DNS resolving the IP address of the peer. The HIP proxy extracts the information of the peer from the DNS response message. When the connection initiator gets the peer information it can proceed in two different ways: It can either send a non-HIP connection initiation packet, e.g. TCP SYN, or an I1 packet. If the Initiator sends an I1 packet, the optimal response for the HIP proxy would be to do nothing with the packet, just let it go through. The I1 packet would be routed to the peer, which in turn would respond with an R2 packet. The HIP base exchange would continue and, once completed, HIP communication could commence between the two peers. What the HIP proxy does when receiving an I1 packet is implementation dependant. The optimal solution has been presented here, but there are other possibilities. The proxy might e.g. try to establish a HIP connection towards one, or both, of the nodes. This would not be a good solution since it requires the HIP proxy to do much superfluous processing.

The connection establishment method is similar regardless of which of the nodes acts as the Initiator. If the initiator decides to use a non-HIP connection initiation packet, the connection setup would proceed as described in Section 4.2.2.1 or Section 4.2.2.2 respectively. However, this solution is not as good as when the Initiator sends an I1 packet. If the Initiator decides to use HIP, the HIP proxy does not need to do much processing. Conversely, if the initiator uses a non-HIP connection initiation packet, the HIP proxy has to set up a HIP connection between itself and the peer node, and it has to solve the puzzle. The best solution would be to burden the HIP proxy as little as possible. Thus a HIP connection between the two HIP nodes would be the preferred option. The preferred solution is shown in Figure 19.

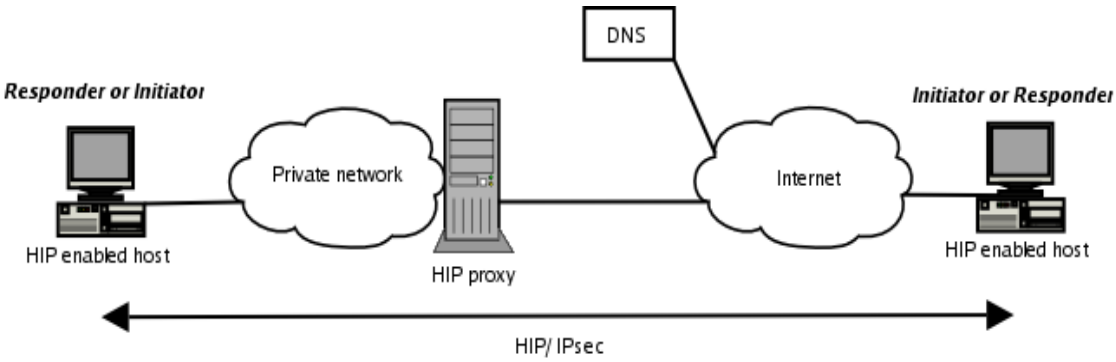

Figure 19: Connection between two HIP enabled hosts with a HIP proxy

4.2.2.4 Legacy host in private network – Legacy host in public network

With legacy hosts in both ends of the connection, the HIP proxy does not provide any real service for the connection. Regardless which of the two nodes initiates the connection it will always be an IP connection all the way. The proxy serves only as a router between the two nodes.

4.2.3 Specific functionality of a HIP proxy

Section 4.2.2 describes eight different cases for communication via a HIP proxy. Detailed solutions for all of them will not be presented in this thesis. In the context of this thesis the private network discussed in the previous sections is a telephone network, GPRS or 3G, and the hosts in that network are mobile telephones. In a mobile telephone network, a packet switched data connection is currently always established by a mobile telephone to a node in the Internet. A connection is never established in the reverse direction. The HIP proxy in a GGSN does not need to be able to handle incoming connection attempts, neither HIP nor non-HIP. These cases are out of the scope of this thesis. The following sections will focus on the cases relevant for this thesis; the cases where the connection initiator is residing in the private network.

4.2.3.1 Connection establishment

There are four relevant cases for establishing a connection via a HIP proxy, namely the ones where the initiator is situated in the private network. Of these four, only one case requires the HIP proxy to take some action, namely when a legacy host contacts a HIP enabled host in the Internet. In the three other cases the proxy should just route the packets between the hosts and the optimal connection for the respective setup will be established. If both hosts are HIP enabled, the connection between them will be a HIP connection. In the other cases the connection will be IP based. A more thorough description of these cases can be found in Section 4.2.2.

The legacy host initiating a connection queries the DNS to resolve the IP address of the peer. The proxy intercepts the response message from DNS and collects the peer information; the IP address and the HIT. When the legacy host initiates the connection establishment the connection initiation packet is routed out from the private network via the GGSN/HIP proxy. The HIP proxy examines the destination address and picks the IP to HIT mappings from an IP-HIT table. If it finds a HIT connected to the destination IP address it puts the received packet on hold, and begins the HIP base exchange with the destination node. After completing the base exchange there is a secure IPsec ESP tunnel between the HIP proxy and the HIP enabled host. The HIP proxy can now send the packet received from the legacy host through the tunnel to the peer host. From this point onwards all traffic between the legacy host and the HIP enabled host is sent through the IPsec ESP tunnel between the HIP proxy and the HIP enabled host. The connection establishment is depicted in Figure 20.

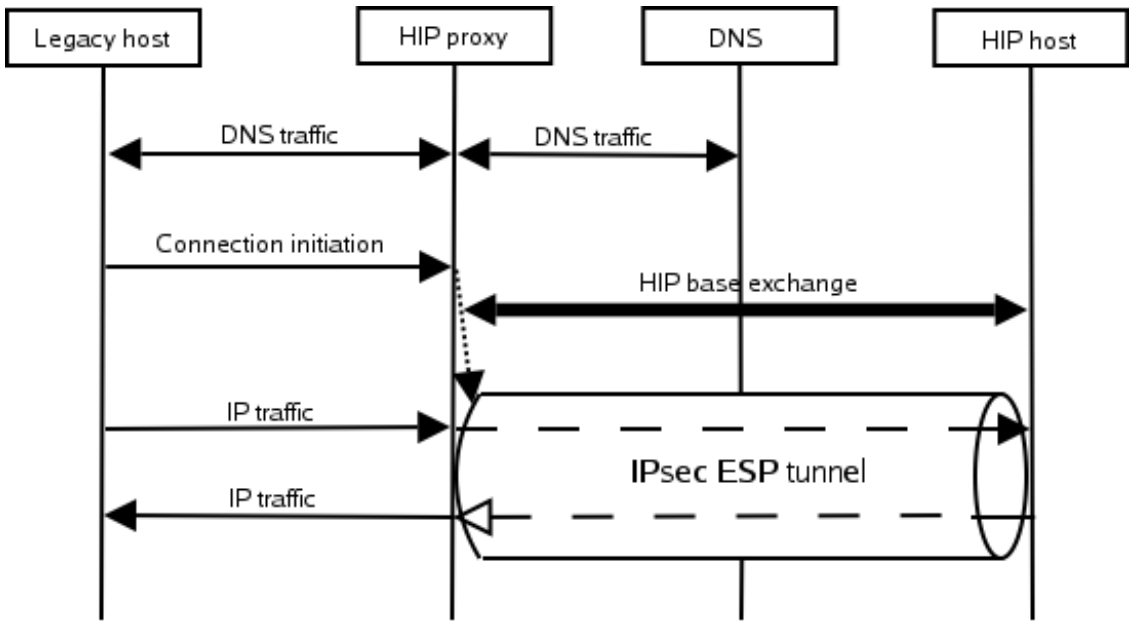

Figure 20: Connection establishment via a HIP proxy

4.2.3.2 Regular traffic

Once the HIP connection has been established between the HIP proxy and the HIP enabled host the two communicating nodes can begin exchanging messages. During this phase of the connection, the tasks of the HIP proxy are quite simple. Incoming packets are converted from ESP packets to IP packets or from IP packets to ESP packets depending on the direction.

Packets originated from a legacy host in the PLMN must be examined. If the destination address of the packet can be found in the HIT-IP table of the HIP proxy, it can be concluded that the packet is destined to a HIP enabled host. In this case the HIP proxy uses the resolved HIT-IP mapping of the destination and the HIT-IP mapping of the sending host to find the SA information that matches the connection. The packet is processed based on the found SA. If no SA is found a HIP base exchange is initiated as presented in Section 4.2.3.1. If the destination address of the packet cannot be found in the HIT-IP table of the HIP proxy, the destination host is not HIP enabled and the HIP proxy just forwards the packet towards its destination.

Packets arriving from the public network are either ESP protected or unprotected. The HIP proxy forwards unprotected packets towards the destination inside the PLMN. In the case that the peer is a HIP host, incoming packets are IPsec protected and they are processed at the proxy. With the help of the SPI of the packet the HIP proxy finds the correct SA, decrypts the packet and updates the address information in the packet. The new IP packet will then be sent to its destination.

4.2.3.3 HIP specific packets

In addition to the four base exchange packets, the HIP draft [11] specifies four other HIP packets; the rekeying packet, the bootstrap packet, the certificate packet and the payload packet. Except for the payload packet that contains data destined to the true end-point of the connection, all the packets terminate at the end-point of the HIP connection. When the HIP proxy receives any of these packets it grabs the packet and processes it. The exception is the payload packet. When the HIP proxy receives a payload packet it should convert the packet to a regular IP packet and send it to the host in the PLMN.

4.2.4 Specific functionality of a HIP proxy in a GGSN

The number of connection scenarios is the biggest difference between a HIP proxy in the Internet and a HIP proxy in a GGSN. When the HIP proxy is located in a GGSN it should never receive packet switched data connection attempts from nodes outside the PLMN (based on current PLMN functionality), i.e. from the Internet. Although this is the situation now, the situation can change in the future. On the other hand, if a HIP proxy is between e.g. a private LAN and the Internet the connection initiator might reside on either side of the HIP proxy. Besides the different connection scenarios, a HIP proxy is very similar regardless of where it operates. The adjustments that need to be done because of different operating environments are mainly focused on the node in which the HIP proxy is planned to operate and not on the proxy itself.

4.2.4.1 Specific functionality of a GGSN with a HIP proxy

The HIP proxy implementation in a GGSN affects the operation of the GGSN. It must know which actions require the participation of the HIP proxy. Basically, this includes all actions of receiving and sending packets: everything from Context Activation and the HIP base exchange to HIP specific packets and regular traffic.

When a legacy UE wants to initiate a connection to a node in the Internet, the GGSN receives a Context Activation request for the UE. The GGSN sends a reply after consulting the HIP proxy; the legacy UE needs to be assigned an asymmetric key-pair (a HI and a secret key) and a HIT. Instead of the HIP proxy, it could as well be the GGSN that assigns the HITs and the keys. The important thing is that the HIT-IP mappings are stored in a place where the HIP proxy can access them; the HIP proxy is the primary user of the HIT-IP mappings. Consequently, the logical choice is that the HIP proxy generates and assigns the HITs and the keys. A GGSN with a HIP proxy may also send back modified Context Activation replies; depending on the implementation, the IP address sent back to the UE might in fact be the HIT that is assigned for that particular UE.

When a packet arrives from the UE to the GGSN it is directed to the HIP proxy function before it is sent out. If there is no SA associated with the connection, the HIP proxy must perform the HIP base exchange with the connection peer. After setting up IPsec SAs, the packet can be sent out from the PLMN using the created SAs to protect the traffic. Alternatively, if the received packet matches an existing SA, the HIP proxy should forward the packet using that IPsec SA. Apart from traffic between two hosts, also replies to DNS queries need to be passed to the HIP proxy. The HIP proxy needs to store the HIT-IP mappings for using the information when initiating a HIP base exchange on behalf of the legacy UE. Depending on the implementation it might also be the GGSN that stores the HIT-IP mappings without the help of the HIP proxy. However, the HIP proxy needs access to the stored information.

The GGSN monitors incoming and outgoing traffic and it must pass packets to the HIP proxy when necessary. HIP specific packets, e.g. UPDATE packets (Section 4.1.4), received from a host in the Internet need some special attention; they need to be handed to the HIP daemon. HIP packets are handled by the HIP connection end-point, so the HIP proxy should not see these HIP specific packets. Other packets, i.e. data traffic, should be passed to the HIP proxy for modification according to the matching SA before they are forwarded to the correct destination.

HIP packets sent by a node in the Internet have the IP address of the GGSN (IP_{GGSN}) as their destination IP address. The destination HIT of the packet is the HIT assigned to the legacy UE, which is the other party of the connection. The GGSN cannot forward such packets anywhere since the destination of the packet (IP_{GGSN}) seems to has been reached. The GGSN must hand over the packet to the HIP proxy, which can operate on it according to the used SA. The source and destination information of the packet is modified and the packet is sent to the legacy UE as a normal IP packet.

5 The HIP proxy, design and implementation

In this chapter the prototype based on earlier specifications is presented. First the used setup will be described including how it represents the theoretical environment described in Section 3.2. The architecture of the implementation is presented, along with its ties to other parts of the system. The implementation itself is looked at, including how it manages connection setup, packet processing, and co-operation with other software and hardware units.

5.1 Implementation architecture

The goal of the work was to design a HIP proxy for a GGSN. The proxy would serve legacy UEs, providing them with the advantages of HIP for the part of the connection that goes over the Internet. The primary benefit the UEs would gain from the use of the proxy would be security for the traffic going through the Internet, which is considered to be the most vulnerable part of the connection. This, naturally, requires that the peer of the connection is HIP enabled. This scenario was depicted earlier in Figure 2 on page 5.

The objective of the thesis is described by the setup presented in Figure 2. However, the prototype implementation that will be presented in this chapter is not built on that setup. Instead, all the nodes used for designing, testing, and running the implementation are personal computers. In the next two subsections the used environment and the software architecture will be presented.

5.1.1 Hardware environment

The hardware setup of the implementation differs from the target setup because it was not practical to use a real GGSN for this work. A personal computer based prototype

gives enough information about the feasibility of the proposed system. Using PCs, it can be shown that the concept of a HIP proxy, be it in a 3G network or somewhere else, can be realized in practice. The network setup used for the prototype implementation of the HIP proxy is shown in Figure 21.

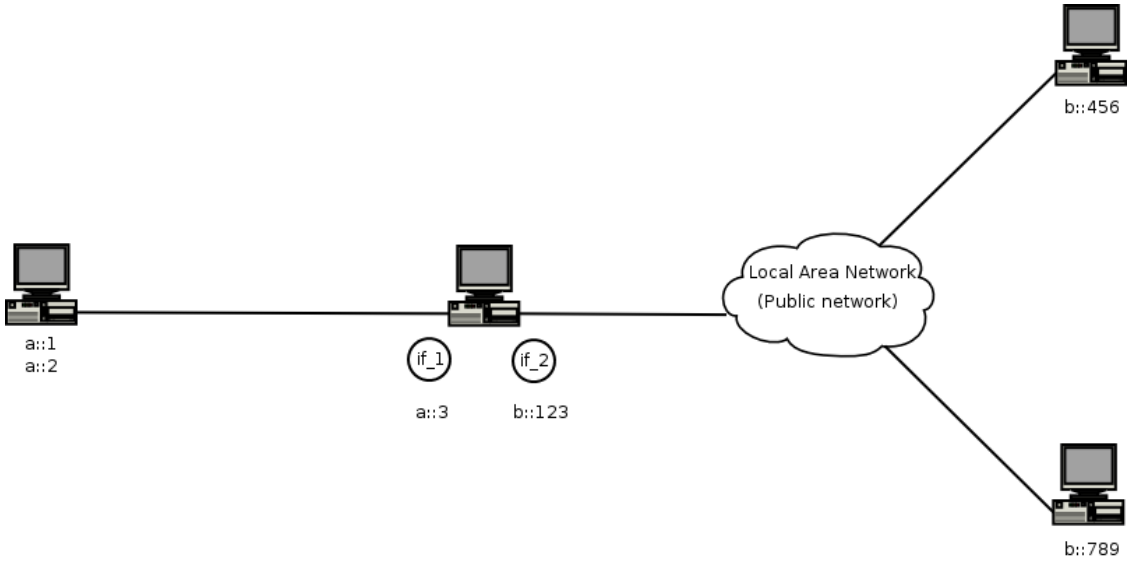

Figure 21: The used network setup

The test network uses IPv6, which is the preferred version with HIP. The addresses displayed in the picture are chosen to be simple to clarify the system. The legacy host is directly connected to the proxy machine via one of its network interfaces (if_1). There are two different IPv6 addresses configured for the legacy host for simulating two legacy hosts connected to the proxy. The HIP proxy has another network interface (if_2), which is connected to the LAN where the two HIP enabled hosts reside. Different network prefixes are used for the two networks that are connected to the HIP proxy. This simulates the existence of the two different networks: the private (a::) and the public (b::) network, e.g. the Internet.

Using different network prefixes for the two networks does not affect HIP communication. Even if the address space used in the private network is private, connections from it to the public network are possible if the proxy knows the HIT-IP mapping of the host in the public network. In this case the proxy modifies packets and uses its own IP address to represent the whole private network. However, there is a

problem if a legacy host wants to communicate with a, for the proxy, unknown host in the public network. Since the proxy only modifies packets for connections for which it has HIT-IP mappings, packets for the unknown host will exit the proxy with a source address belonging to the private address space. To get the response packets routed correctly, the host in the public network needs to be explicitly informed where to send packets with the destination in the private network. This can be done using the *route* [50] command. A more elegant solution would be to perform NAT on packets that do not utilize the HIP proxy.

5.1.2 Software environment

The choice of operating system for the nodes in the system was made based on the available support for HIP. The HIP prototype developed at Ericsson Research in Finland [51] is made for FreeBSD [52] making it a natural selection. The operating system on all of the nodes is FreeBSD version 5.2. The legacy host uses version 5.2-RELEASE, the rest of the nodes use slightly modified versions of FreeBSD 5.2.1-RELEASE-p4. The nodes that require the HIP enabled IP-stack, namely the HIP enabled hosts and the HIP proxy, use the HIP modified source code developed at Ericsson.

5.1.3 Preparations for the HIP proxy

The HIP proxy is an application handling HIP traffic on behalf of legacy hosts. The packet processing on the host running the HIP proxy application must be modified so that the proxy receives all traffic that goes through the host. The proxy operates on the received packet, e.g. changing the IPv6 addresses to HITs, before injecting it back to the packet handling queue of the host. FreeBSD provides a suitable method for diverting incoming packets away from normal packet processing, namely the divert [53] socket [54]. However, the divert protocol is not IPv6 compatible, it only works with IPv4. In the following paragraphs there will be some references to source files of FreeBSD, they can be obtained by acquiring FreeBSD or be browsed online [55].

Our HIP proxy prototype uses only pure IPv6, so we had to make an IPv6 friendly version of the divert protocol; divert6. The source code for the original divert functionality, located in *netinet/ip_divert.c*, and our IPv6 version (*netinet6/ip6_divert.c*) differ mainly in the used data structures. Instead of using IPv4 header structures we use IPv6 header structures, and instead of the IPv4 socket address structure we use a slightly modified IPv6 socket address structure. Similarly, we use IPv6 specific functions instead of the IPv4 specific functions when applicable. The reason for all these changes is that IPv4 and IPv6 have different header structures and address length. The changes made to the IPv4-version of divert are all quite straightforward. Only one functional change had to be done to the function div6_output() compared to the IPv4 counterpart, div_output(); the IPv6 version always sends the packet to ip6_output(), while the original IPv4 version usually sends it back to ip_input(). This way the IPv4 version makes the divert functionality almost invisible, while the new IPv6 version acts like a bridge between the ip6_input() and the ip6_output() functions.

In addition to the base divert system changes some other modifications are also required. The packet handling functions ip6_input() in *netinet6/ip6_input.c*, ip6_output() in *netinet6/ip6_output.c*, and ip6_forward() in *netinet6/ip6_forward.c* check if there is a firewall rule that says that the packet should be diverted. If there is, they abandon the packet. This needs to be changed in all these files, even if the packets that use divert6 will probably never reach the ip6_forward() function.

The ip6_input() and the ip6_output() functions need some additional editing to get them aware of the new divert6. To get the input function to use divert6 when appropriate, the matching firewall rule and the divert port need to be resolved. After sanity checks have been performed on the packet it can be diverted if indicated by the firewall rule. The output function only needs one additional modification; in the beginning of the function the data added by the divert6 functions has to be peeled off the packet. To get the operating system to use the divert6 protocol, it has to be added to the IPv6 protocol switch in *netinet6/in6_proto.c*. The file contains the definitions for other protocols, e.g. IPv6, which can be used as a template when adding divert6.

5.1.4 Configuring the firewall

In our implementation the diverting of packets is realized using the firewall and divert sockets for IPv6. In FreeBSD, we use the ip6fw firewall [56]. Below, table 2 shows an ordered list of the needed firewall rules. The order of the rules is important because packets are compared against the rules until a matching rule is found.

Table 2: Firewall rules

1. allow ipv6 from 4000::/2 to 4000::/2 out
2. allow 99 from any to any
3. allow ipv6-icmp from any to any icmptype 135,136
4. allow ipv6 from any to any via lo0
5. allow ipv6-icmp from :: to ff02::/16
6. allow ipv6-icmp from fe80::/10 to fe80::/10
7. allow ipv6-icmp from fe80::/10 to ff02::/16
8. allow ipv6 from any to ff02::/16 via <interface (if_1)>
9. allow ipv6 from a::/16 to a::/16
10. divert 22222 ipv6 from a::/16 to any via <interface (if_1)>
11. divert 22222 ipv6 from 4000::/2 to 4000::/2
12. divert 22222 ipv6 from any to a::/16
13. allow all from any to any
14. deny ipv6 from any to any

The first rule allows all outgoing traffic with IP addresses having a two-bit prefix of 01_{bin}, identifying packets as having HITs as addresses. This way the packets modified by the proxy will not be interrupted during their handling in the output routine, where they will be converted back to packets using IP addresses. Rule number two allows passing of traffic that uses protocol number 99, the number assigned to the HIP protocol in the used HIP implementation. All HIP specific packets will get through

uninterrupted. Rule 3 allows all IPv6 ICMP [57] neighbor solicitation and advertisement packets.

The firewall rules 4, 5, 6 and 7 define how to react to link-local and broadcast traffic. These are not directly relevant for the HIP proxy but are used for IPv6 traffic in general. Likewise, rule 14 is not there to facilitate the work of the proxy. It is the basic rule for any firewall, denying all unspecified traffic. Rule 13 is used for testing purposes, overriding rule 14 completely. Rule 8 allows all broadcast traffic to and from the private network, while rule 9 allows traffic between hosts in the private network.

The rules 10-12 are used for defining which packets are relevant for the HIP proxy and should be diverted to it. The packets matching any of these rules are all diverted to port 22222, which the HIP proxy is listening to. First, rule 10 says that all traffic from the private network should be diverted: all packets with a destination address belonging to the private network will be diverted to the HIP proxy, except if the packet has matched any of the previous rules. Rule 11 states that all packets with HITs as addresses will be diverted. This rule is used when receiving packets on a HIP association, after the received packet has gone through the IPsec handling. A packet received from the public network, but not over an SA, is diverted based on rule 12. Packets diverted as a consequence of rule 12 are forwarded to the private network by the HIP proxy. Rule 13 allows all traffic to and from the HIP proxy, making it possible for packets to leave the host. The rule also makes it possible to contact the host remotely, e.g. when the HIP proxy administrator needs to modify the configuration of the proxy. However, rule 13 is dangerous in a real network environment and should be replaced by an advanced set of rules that gives better security.

5.1.5 How the HIP proxy fits in the picture

So far we have only talked about entities that make the use of a HIP proxy possible. In this section we will look at how the proxy is tied to all of these elements. The operation of the HIP proxy will be discussed in Section 5.2.

5.1.5.1 The HIP proxy and outgoing packets

The HIP proxy, residing between the private and the public networks, is considered to be a part of the private network. Thus, traffic originating from the private network towards the public one is said to be outgoing traffic. Figure 22 below depicts how the HIP proxy co-operates with various entities in the host while processing outgoing packets.

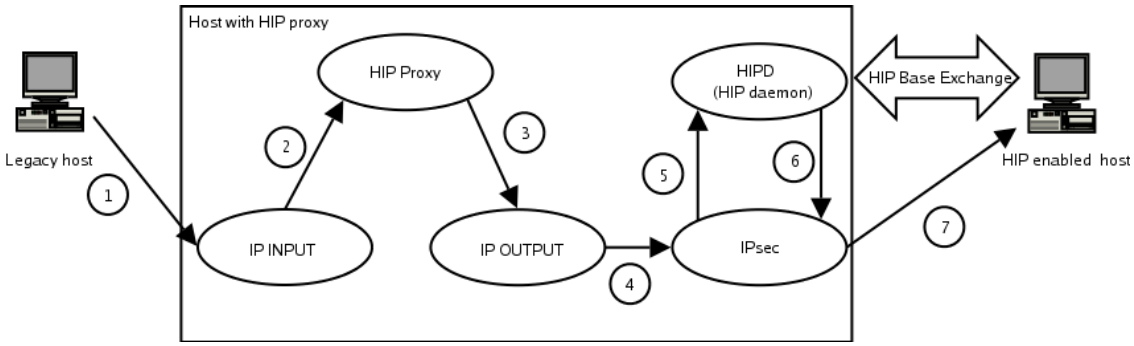

Figure 22: HIP proxy processing of packet from the private network

In step 1, an IP packet arrives from a legacy host in the private network to the proxy. The destination address of the packet belongs to an HIP enabled host in the public network. During the handling of incoming packets a firewall rule (rule 10) states that the packet should be diverted. The packet is handed over to the HIP proxy in step 2. The HIP proxy browses its IP-to-HIT mappings for a match for the source and destination addresses of the packet. Upon finding a match for both addresses, the IP addresses are exchanged for the matching HITs, and the packet is sent to output handling in step 3. In the event that a mapping cannot be found, the packet is sent unmodified to the destination, via the output handling.

Now, in step 4, the packet is handed over to IPsec processing. If no SA is found for the connection between the source and destination HITs, the HIP daemon is signaled in step 5, and it performs the HIP base exchange between the proxy and the HIP host. This results in SAs being created for the connection. The IPsec processing can now continue, step 6, and in step 7 the encrypted packet will be sent out, with the HITs

replaced by IP addresses for routing in the network. When there already are SAs for the connection, steps 5 and 6 are skipped.

5.1.5.2 The HIP proxy and incoming packets

The HIP proxy is not interested in traffic from the public network that is neither HIP packets nor IPsec packets. These plain IP packets are just forwarded to the private network. This section describes how a packet passes through the HIP proxy when it is sent on a HIP association from a HIP enabled host to a legacy host in the private network. In case there is no SA created for the connection the HIP proxy will just function as a router and forward the packet. Figure 23 presents the processing of incoming packets.

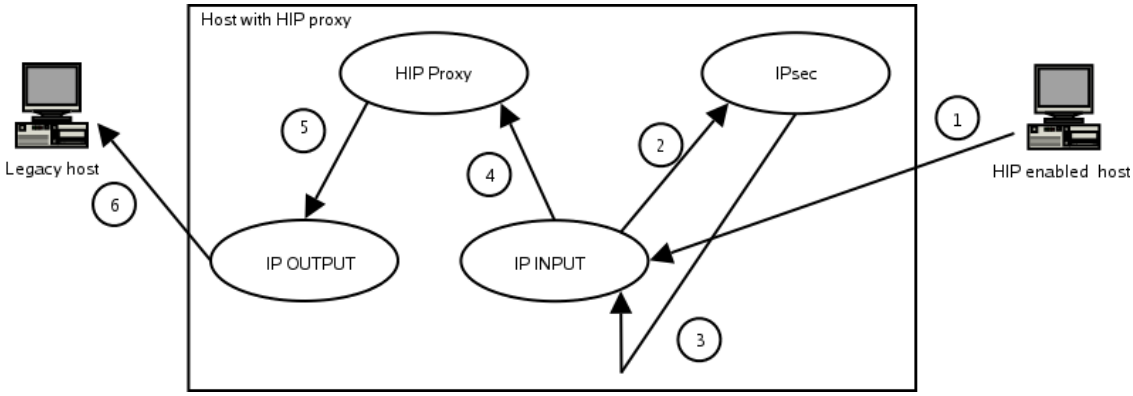

Figure 23: HIP proxy processing of packets from the public network

In step 1, a packet arrives over an SA to the incoming packet handling of the HIP proxy. The received packet is in step 2 handed to IPsec processing. If the packet is a HIP specific packet, e.g. an UPDATE packet, the packet is instead handed over to the HIP daemon for further processing. In step 3, the packet processed by IPsec is injected to the input handling again, now with the IP addresses replaced by HITs. The packet will match a firewall rule (rule 11) that diverts the packet to the HIP proxy in step 4. The HIP proxy examines its HIT-to-IP mappings and replaces the source and destination HITs of the packet with the matching IP addresses. In step 5 the HIP proxy

sends the packet to output processing, and finally in step 6 the packet is sent to the destination node in the private network.

5.2 Inside the HIP proxy

The basic functionality of the HIP proxy application closely resembles the functionality of a NAT. While a NAT changes only the address of the host in the private network, the HIP proxy changes both the addresses when packets pass it. After the possible IPsec processing of the packet, the final result is similar to what would be expected of a NAT; a packet leaving the private network has a new source address and a packet entering the private network has a new destination address. In the following subsections the data structures used by the application are presented and the application architecture is examined. After that we will advise on how to use the HIP proxy.

5.2.1 The used data structures

The most important data for managing connections is the IP-address-HIT mapping information. This information is used for making the IP address to HIT translations, or vice versa. The HIT-IP mappings are stored in two linked lists. These lists contain structures consisting of an IPv6 address, the matching HIT, and an indicator giving the status of the structure. One of the lists is for information of HIP enabled hosts in the public network that can be contacted via the HIP proxy. The other list is for legacy hosts in the private network, with the HIT in each structure being the HIT appointed to the host by the HIP proxy administrator. Because both lists contain the same kind of structures one could manage with only one list. However, from the implementation and performance point of views the use of two different lists is justified. With two lists it is easier to determine whether a packet is going to, or coming from, the private network. Also finding the matching mappings is faster. The two linked lists are an essential part of the HIP proxy and they are referenced each time the proxy receives a packet. There is no need for any other special data structures. The implementation reads a

configuration file to maintain the information contained by the two lists. There will be more on the configuration file in Section 5.2.3.

5.2.2 Application structure

The presentation of the application structure will follow the pseudo code representation of the HIP proxy, shown in Table 3.

Table 3: Pseudo code representation of the HIP proxy

```
#01   BEGIN
#02
#03     CREATE divert socket
#04     READ configuration file
#05     CREATE hip contexts
#06
#07     /* BEGIN read/write loop */
#08     WHILE true
#09
#10       READ diverted packet
#11       DETERMINE hit-to-ip mappings
#12
#13       IF mappings found THEN
#14         GOTO send_packet_now
#15
#16       IF direction of packet = out THEN   /* out = to public network */
#17         SWAP ip for hit
#18
#19       ELSE    /* IF direction = in, in = to private network  */
#20         SWAP hit for ip
#21
#22       DETERMINE new checksum for packet
#23
#24     send_packet_now:
#25       WRITE packet
#26
#27       IF update required THEN
#28         READ configuration file
#29         CREATE hip contexts
#30
#31     /* END read/write loop */
#32   END
```

The application needs to create a socket for receiving and sending diverted packets. The socket is bound to the same port that is used in the firewall rules for diverting

packets, in this case 22222. After establishing the socket, the proxy reads its configuration file, line 4 in Table 3, and adds the entries to the two linked lists as mentioned in Section 5.2.1. Before beginning to handle packets, the HIP proxy creates HIP contexts.

A pre-requisite for performing a HIP base exchange between two hosts is that the Initiator has a HIP context matching the desired connection. The context contains the HIT of the Initiator and the IP address and the HIT of the Responder. The HIP proxy creates all the possible HIP contexts in advance, before they are needed. This means that for each legacy host whose information was found in the configuration file, it creates a context towards each HIP enabled host defined in the configuration file. When all contexts are created, the application enters an infinite packet-handling loop and begins waiting for diverted packets.

The firewall function diverts received packets to the HIP proxy when they match one of the divert rules in the firewall. Upon receiving a packet, on line 10, the HIP proxy inspects the source and destination addresses of the packet. On line 11, the addresses are compared to the IPv6 addresses and the HITs stored in the two linked lists. When receiving a packet, there are only two scenarios that lead to actual processing by the HIP proxy. In the rest of the cases, the packet is only forwarded:

- A match for both the source and the destination address is found in the same list. This means that the connection peers are in the same network. When this happens, the HIP proxy should not be involved and the packet is forwarded.
- One of the addresses in the packet is a HIT and the other is an IP address. In this case the proxy cannot do any processing on the packet; there cannot be traffic that has this format. The packet is forwarded, but could just as well be dropped.
- Traffic that would require the HIP proxy to establish a HIP association with a node in the private network is directly forwarded by the proxy. HIP is not needed in the private network. This is the case when receiving a packet from the public network, with a destination IP address of a host found in the private

network. However, the proxy should check to see if the packet is an I1 packet and the destination IP address is configured into the proxy. In this case it is the host in the public network trying to perform the HIP base exchange with a legacy host. The proxy should answer to this packet with an R1 packet, and establish SAs.

- Either, or both, of the addresses cannot be found in one of the lists. The packet must be forwarded. An attempt to perform opportunistic mode HIP could be made, but the current prototype implementation does not do this.

If the source address of an incoming packet equals to a HIT found in the HIP host list and the destination is a HIT found in the legacy host list, we know that the packet is on its way to the private network. Because the packet contains HITs we also know that the packet was received over an SA, and we know the IPv6 addresses matching the HITs. Since we have found mappings for both addresses, one from each list, we can conclude that the proxy should process the packet. This is the case when the proxy receives traffic over an HIP association, from a HIP enabled node in the public network.

The other scenario that leads to processing by the proxy is if the source address equals an IPv6 address found in the legacy host list, and the destination is an IPv6 address found in the HIP host list. This happens when a legacy host sends a packet to a HIP enabled host known by the proxy. All other possibilities are unacceptable and lead, via lines 13 and 14 in Table 3, to forwarding the packet unchanged. One should note that HIP specific packets never reach the proxy since they terminate at the HIP daemon [11].

A map of all the possible traffic that the proxy can receive is shown in Table 4. The cases that result in processing by the proxy are marked with “OK”. The cases that do not result in processing by the proxy are marked with “Not OK” and an explanation of why they do not qualify. A more detailed description of the reasons why a packet might not qualify for processing was presented earlier in this section.

Table 4: The possible connection scenarios for a HIP proxy

Source / Destination	the **Legacy host list**	**HIT**, found in the **HIP host list**	**IPv6** address, found in the **Legacy host list**	**IPv6** address, found in the **HIP host list**	**Unknown**
HIT, found in the **Legacy host list**	Same list **Not OK**	OK **Process**	One HIT, and one IP **Not OK**	One HIT, and one IP **Not OK**	Unknown source **Not OK**
in the **HIP host list**	HIP traffic cannot come from legacy host **Not OK**	Same list **Not OK**	One HIT, and one IP **Not OK**	One HIT, and one IP **Not OK**	Unknown source **Not OK**
IPv6 address, found in the **Legacy host list**	One HIT, and one IP **Not OK**	One HIT, and one IP **Not OK**	Same list **Not OK**	No HIP to public network **Not OK**	Unknown source **Not OK**
IPv6 address, found in the **HIP host list**	One HIT, and one IP **Not OK**	One HIT, and one IP **Not OK**	OK **Process**	Same list **Not OK**	Unknown source **Not OK**
Unknown	Unknown destination **Not OK**	Unknown destination **Not OK**	Unknown destination **Not OK**	Unknown destination **Not OK**	Unknown destination and source **Not OK**

The HIP proxy finds out the direction where the received and processing requiring packet is going by verifying the lists from which the source and destination addresses are found. If the source address can be found in the legacy host list, and the destination address in the list of HIP enabled hosts, the proxy concludes that the packet is on its way out from the private network. Conversely, if the addresses are found in the opposite lists, the packet is on its way into the private network. Based on this information the addresses of the packet are now switched as appropriate; IP addresses are switched for HITs, or HITs for IP addresses. This corresponds to lines 16-20 in Table 3.

Modifying the addresses of the packet breaks the checksum included in the packet. On line 22, the proxy sets the checksum to zero and recalculates it. The location of the checksum depends on the upper layer protocol type. When the checksum has been recalculated, the packet can be sent to output processing; line 25 in Table 3. The final step of the read/write loop is to re-read the configuration file for updating the entries in

the two lists, and create new HIP contexts when necessary. When to re-read the configuration file is discussed in more detail in the next section. The packet-handling loop has now been executed so the program moves to waiting for the next packet to handle.

5.2.3 Configuring the HIP proxy

The configuration information for the HIP proxy is stored in a configuration file. The file is a text file that contains the IPv6 address and HIT pairs for hosts, one pair per line. The syntax of the file, along with an example line, is shown in Figure 24.

Syntax:	**<Selector> <HIT> <IPv6 address>**
Example:	H 4432:15e8:bba7:cd84:a0aa:5ab4:d82d:73bd 2001:0:0:0:0:0:0:1

Figure 24: Configuration file syntax

The selector is a character identifying which list the pair belongs to. If the selector is the character ‘L’, as in legacy host, the pair belongs to a host in the private network. A pair belonging to a HIP enabled host is marked with the character ‘H’. Lines beginning with the ‘#’ character are interpreted as being comments and are ignored when the configuration file is read.

Following the pseudo code representation of the HIP proxy, in Table 3, one can see that the configuration file is read once before the read/write loop, and then again when necessary. The re-reading is done to enable changes in mapping information without the need to restart the proxy each time an entry is changed in the file.

The problem with using a configuration file is to know when to re-read it. It would be desirable that changes to the configuration file take effect immediately. The use of a configuration file for managing the host information is sufficient for testing the prototype implementation. The method used here for managing updates is to re-read

the configuration file after a certain amount of packets have passed through the HIP proxy. For testing, a value of 10 has been used. but that is a too small number to be used in practice. The value is too small, because the process of re-reading the file and updating the two lists is somewhat complex, and takes time. The choice of a good value depends mainly on the amount of hosts in the file. With many hosts, reading the file takes longer. The amount of hosts also affects the amount of traffic; more hosts means more traffic. In a real-life environment reading a configuration file is not necessarily a good solution, but some other method should be used

In addition to an IPv6 address and a HIT, the structures in the two lists contain an indicator that indicates if the entry is a new one. When an entry is read from the configuration file and added to one of the lists, it is marked as being a new entry. Before it is added to one of the lists, the list is checked for a matching entry; if there already is an entry corresponding to the new one, the new entry is discarded. When creating the HIP contexts, only HIP contexts for the new entries are created. This is accomplished with the help of the indicator. When all the necessary contexts have been created the indicator for each entry is changed to indicate that the entry is old.

5.2.4 Using the HIP proxy

The main task of the HIP proxy is to replace IP addresses with HITs, or vice versa. The IP address to HIT mappings are read from the configuration file, which is edited by the HIP proxy administrator. The asymmetric key pairs, the HIs and the matching secret keys, and the corresponding HITs are generated by the proxy administrator and made available for the HIP daemon.

Before the HIP proxy application can be launched, the HIP daemon needs to be started on the machine that will run the proxy. If the HIP proxy is started before the HIP daemon, the HIP contexts that the proxy uses will not be created since the process of creating contexts requires the use of the HIP daemon. A proper configuration file should also be available for the proxy so that it can obtain information about its clients.

6 Analysis

In this chapter the prototype implementation, presented in Chapter 5, is analyzed. The solution is evaluated based on the evaluation criteria stated in Chapter 3. Some problems found during implementation are identified and explained. Finally also future work and improvements to the implementation are discussed along with expectations of the usefulness of the solution.

6.1 Evaluating the solution

Before we could evaluate the solution, we needed to know that the HIP proxy actually does what it is supposed to do. Even if the packets seem to be processed by the proxy, and there is traffic between the private and the public network, we must still be sure that HIP is being used when appropriate. This can be achieved by using tools such as Ethereal [58] and tcpdump [59]. With these tools, the network traffic to and from the HIP proxy can be inspected.

Using Ethereal we verified that the HIP proxy performs the HIP base exchange when there is no SA between the proxy and the HIP enabled destination. We also confirmed that packets received from the private network leave the proxy as ESP packets. In addition to the mentioned tools, we also analyzed the debug output from both the HIP daemons and the HIP proxy. In verbose mode the HIP proxy writes debug information on each packet it receives and sends. The information includes source and destination addresses of the packet as well as some protocol specific information.

6.1.1 Testing the implementation

The testing requires data traffic between the end-hosts. We used ping6 [60] as a simple application to test that we get traffic through to the public network. We selected ping6 as a test program since it is easy and fast to analyze. When we were convinced that the

proxy was processing the traffic properly, we switched over to using SSH [61]. SSH is more useful than ping6 as a test program since it generates traffic that can be considered to be "normal Internet traffic"; the traffic is not just different with respect to content, but SSH also generates a different flow of data compared to ping6. Even if the services provided by SSH are of no interest here, it is still a good representation of a standard network program used on a host residing in the private network.

Both ping6 and SSH worked very well with the HIP proxy. Sometimes a packet or two was lost because of the delay resulting from performing the HIP base exchange, but besides that everything worked smoothly. We also had two connections running simultaneously, from the two addresses configured to the legacy host to the two HIP enabled hosts residing in the LAN. This proved that the HIP proxy can handle multiple SA pairs. The difference between a HIP end-host and HIP proxy is that the end-host handles (at the moment) only one HI, which is its own identity. The proxy needs to be able to use multiple local identities.

6.1.2 Evaluation of the solution vs. the criteria

The primary goal was to implement a HIP proxy that would function in a GGSN. Even if the prototype implementation is not done for a GGSN, it does perform the functions required by a HIP proxy in a GGSN; when a node in the private network initiates a connection, the proxy establishes HIP associations between itself and HIP enabled node in the public network. In Chapter 3, some evaluation criteria were presented. We will now inspect how well our implementation fulfills these criteria.

6.1.2.1 Security

The first two criteria were regarding the security of the HIP proxy and the configuration tool. The security of these two entities relies heavily on the security of the node on which they are located. Thus the security is not directly HIP proxy specific, but node specific. Since we were running the HIP proxy on an up-to-date

version of FreeBSD, it can be considered to have been at least somewhat secure; still, one cannot give any guarantees. To get the security to an acceptable level there should be a security policy stating how the host can be accessed and who are authorized to access the host. Letting only the HIP proxy administrator access the host, and limiting the access methods to secure ones, e.g. SSH and HIP, the HIP proxy can be deemed to be secure.

6.1.2.2 Performance

The next criterion was that the proxy would perform well. The proxy was run on a "normal" laptop computer with a 2,26GHz mobile P4 processor and 512MB RAM. To test what kind of delays the use of the HIP proxy causes, we used ping6 to get measurements for the round-trip time of the connection between the legacy host and the HIP host. We used the following test cases:

- Ping HIP enabled host from legacy host, without proxy
  - One and two simultaneous connections
- Ping HIP enabled host from legacy host, using proxy but not using HIP
  - One and two simultaneous connections
- Ping HIP enabled host from legacy host, using proxy and using HIP
  - One, two, four and eight simultaneous connections

This way we got values to analyze to resolve how the use of HIP and the proxy affects the connection. The average round-trip times, taken over 20 packets, for the different cases are presented in Table 5. The bold figure is the average of the measured average round-trip times, which are presented in parenthesis. It should be noted that the cases in which HIP is used, the SAs existed prior to using ping6. This was done so that the HIP base exchange would not affect the measured values. The base exchange adds delay to a HIP connection, and it cannot be removed or reduced as long as HIP is used.

Table 5: Round-trip times for the connection

Using proxy	Using HIP	Connections	Average round-trip time for 20 packets
No	No	1	**0,624ms**
No	No	2	(0,597ms; 0,635ms) **0,616ms**
Yes	No	1	**0,698ms**
Yes	No	2	(0,678ms; 0,690ms) **0,684ms**
Yes	Yes	1	**0,851ms**
Yes	Yes	2	(0,822ms; 0,841ms) **0,832ms**
Yes	Yes	4	(0,793ms; 1,692ms; 0,848ms; 0,825ms) all: 1,040ms; without 1,692ms: **0,822ms**
Yes	Yes	8	(0,831ms; 0,877ms; 0,879ms; 0,924ms; 0,812ms; 0,833ms; 0,867ms; 0,949ms) **0,872ms**

The first four cases found in the table show us that having two simultaneous connections instead of one, regardless if the HIP proxy is running or not, does not significantly affect the round-trip times. In these cases HIP is not used and the proxy does not have to do any HIP processing. As the test results show, running the HIP proxy adds some 12% to the round-trip time. The reason for this is that instead of directly forwarding the packets from input handling, they are diverted to the proxy and then sent to output handling. The two different ways of forwarding the packets take a different amount of time. Still, the difference is not very big, and it is quite acceptable since the RTT value stays in the same order of magnitude. The last four rows show that with only a few connections the round-trip time when HIP is used stays fairly constant. But, HIP does induce some delay for the traffic; approximately 35% longer round-trip times compared to neither using HIP nor the HIP proxy, and roughly 22% longer compared to using the proxy but not HIP. The added delay is still relatively small and results from using the cryptographic functions that provide security for the connection. One cannot have both speed and security since security requires computation, with the amount of computation depending on the used protocols. Further testing is needed to determine the effects on the round-trip time of using the proxy. Especially tests with hundreds or even thousands of simultaneous connections should be performed. However, this cannot be done manually but some testing tool would be needed.

Another aspect of the proxy's performance to be tested was how the number of hosts configured into the proxy impacts the round-trip time. In the previous tests only three hosts were configured into the proxy. In the following tests both lists were populated with bogus entries that were almost identical with the "real" entries. The bogus entries differed from the real ones only in the last few bits for both HITs and IPs. The entries for the hosts to be contacted were placed last in the lists so that the proxy had to check all the entries before it found what the entries it was looking for. For each packet the proxy had to go through both lists from beginning to the end. The tests were done with 10, 50, 100, 500 and 1000 entries per list and the results of the tests are presented in Table 6.

Table 6: Effects of having many hosts configured into the linked lists

Hosts/list	Average round-trip time for 20 packets
10	**0,676ms**
50	**0,693ms**
100	**0,705ms**
500	**0,730ms**
1000	**0,770ms**

As could be expected, adding hosts to the lists does increase the delay. Moving from 10 to 1000 hosts adds approximately 0,1ms to the round trip time, which is an increase of about 14%. This is not very serious, we must remember that the HIP proxy prototype is not meant for huge networks. When we have 2000 hosts configured into the proxy, the delay from looking up the HIT-IP mappings is not the big problem. A more important problem with that many hosts is the amount of traffic that the proxy has to handle. It is also important to remember that the received results are for a worst-case scenario, so the actual average added delay for the respective case is smaller in real life.

Based on the presented measurements, we can see that the connection does not suffer greatly because of the used security functions. We can also see from Table 5 that increasing the amount of connections from one to eight does not significantly affect the delay. Getting measurements for a larger number of connections would require the use

of a testing tool that generates a lot of traffic. Furthermore, even if the amount of hosts configured into the proxy increases the delay, it is not something that we need to be concerned about at the moment. However, future versions of the proxy will probably have to deal with that problem.

6.1.2.3 Robustness, fault tolerance and stability

The next three criteria, robustness, fault tolerance and stability, are somewhat hard to measure. However, during testing there have not been any problems that would indicate that the program performs poorly in these categories. When the proxy receives a packet that it does not understand it forwards the packet without any modification. If the proxy application gets shut down it does not affect the existing SAs because they are handled by the HIP daemon. Thus, if the proxy application is restarted, the communication can continue.

In an implementation of this kind, the configuration file can cause problems. However, even by inducing syntactical errors into the configuration file, the proxy has handled the situations appropriately; only legit entries are recorded.

6.1.2.4 Complexity

The last criterion was that the implementation should have low complexity. The source code of the application is roughly 650 lines long, of which approximately 25% is comments. The overall structure of the code is not intricate; there are not many nested ‘if’ and ‘for’ statements making the code logically complex, or big and complicated data structures. There is, however, one function that is more complex than the rest of the implementation. This function adds the entries from the configuration file to the two linked lists and creates HIP contexts. The complex part of the function is the creation of all the HIP contexts, which requires some nesting of ‘if’ and ‘for’ statements. However, the function is not overly complex, and has been working as expected.

6.1.2.5 Effects of having the HIP proxy in a GGSN

The evaluation of the solution against the criteria, presented in the previous subsections, is based on our prototype implementation for computer networks. However, if the proxy actually was located in a GGSN the evaluation would differ, the requirements would be more stringent.

A standard GGSN has strict security demands. Adding the proxy functionality to a GGSN will naturally provide security for the HIP proxy through the security properties of the GGSN. The network operators have put a lot of effort not to allow unauthorized access to their nodes. The security features of the GGSN would probably be sufficient for the HIP proxy. However, introducing a new entity to the GGSN might introduce new possible vulnerabilities to the GGSN. When adding the HIP proxy functionality to a GGSN it is important to assure that the HIP proxy is secure, and that the combination of the HIP proxy with a GGSN is secure.

The results obtained when measuring the delays should be sufficient for the proxy. However, the results are useless since the measurements were only done for a few connections. A HIP proxy in a GGSN would need to handle the information of thousands of hosts, and also manage numerous simultaneous connections. Testing a plausible scenario, with e.g. a hundred simultaneous connections, is difficult without some kind of testing or simulation tool and it is difficult to predict how well the proxy would perform. It is obvious that using a configuration file for storing the information of hosts is out of the question and some other method is required.

Current tests give only a vague picture of the robustness, fault tolerance, and stability of the proxy. Requirements for a real product are extremely high. Network operators require products that are running practically all the time with hardly any downtime. This means that faults should not occur or be very rare. Since faults should be rare, it would be desirable that the implementation would be simple - unnecessary complexity might induce errors. However, keeping the system simple is difficult when the proxy needs to handle thousands of hosts simultaneously. E.g. the linked lists used for storing

the HIT-to-IP mappings should not be used as such: With thousands of hosts, browsing the lists for a match takes too long. Instead some sort of hash table might be used. The implementation will become even more complex e.g. if the HLR would be involved in the system to provide static HITs for the legacy UEs.

6.1.3 Unresolved issues

The current HIP proxy implementation fulfills the criteria set for it and works nicely. However, during the implementation and the testing of the proxy some of the in theory identified difficulties were proven to be real problems. Also previously unidentified problems were stumbled upon.

6.1.3.1 Expected problems

The goal was to design a HIP proxy for a 3G network, with the proxy residing in a GGSN. In that scenario, as the system works today, connections are always initiated from the private network, i.e. from the 3G network. However, the actual prototype implementation was done for computer networks, which allow connections to be initiated from either side of the proxy. If used in this kind of an environment, which might be the case in the future of 3G, it would be desirable that the HIP proxy could handle connections initiated from either of the two networks. With both of the communicating parties configured into the proxy this is not possible with the prototype implementation presented in Chapter 5, at least not without some modifications.

When a host in the public network initiates the connection, the proxy forwards the first packet to the host in the private network. This connection scenario is one of the cases marked with “Not OK” in Table 4 in Chapter 5, meaning that the proxy does not process the packet, but just forwards it. The legacy host replies to the packet, and the reply packet is diverted to the proxy. With the host in the public network configured into the proxy this will however match one of the cases marked with “OK” in Table 4. In this case the HIP proxy is required to process the packet; the HIP proxy cannot

distinguish between the first packet of a connection and any other packet sent from the legacy hosts. This leads to the situation where the HIP daemon believes that the packet is the first packet of a new connection. Therefore it initiates the HIP base exchange with the host that originally initiated the connection. The result of this is that the HIP enabled host that initiated the connection now has two connections; one to the HIP proxy and one directly to the legacy host. The HIP enabled host will not use the connection set up by the HIP proxy since it believes that it is between itself and the HIP proxy. The other connection, the one towards the legacy host, is not working since all the replies from the legacy host arrive via the connection set up with the HIP proxy. Thus the HIP enabled host will conclude that the legacy host is unreachable.

The described problem only occurs if the HIP enabled host initiates, regardless of the used protocol, the connection using the IP address of the legacy host as destination address. However, if the initiation packet is an I1 packet, with the destination set to the IP address of the HIP proxy and HIT_R is the HIT assigned to the legacy host, the connection will be established. The resulting connection will be similar to what would be created if it was the legacy host that initiated the connection; a HIP association is used between the proxy and the HIP enabled host.

A similar problem occurs when both the host in the public and the host in the private network are HIP enabled. Regardless of which of the hosts initiates the connection, the HIP proxy will interfere. The HIP base exchange packets will be properly forwarded by the proxy, since firewall rule 2, Table 2, Chapter 5, states that they should be forwarded. However, the first ESP packet sent by the legacy host will be diverted to the HIP proxy, following firewall rule 10, Table 2. When the proxy examines the source and destination addresses, it notices that there are no SAs for the connection. The proxy will initiate the HIP base exchange with the HIP enabled host. Now there will again be the same problem with two connections; packets sent by the host in the public network will use the direct connection to the host in the private network while the replies will be diverted to the proxy and sent on the newly created SAs.

The described problem, which appears in two different scenarios, can be solved quite easily. If the address of either of the hosts is not configured into the HIP proxy, the proxy will always forward the packets without modification. In the case where both hosts are HIP enabled, the problem is solved almost automatically; if a host in the private network is HIP enabled it does not need to be configured into the HIP proxy. Thus, if the proxy configuration file is filled properly, i.e. without the information of the HIP enabled host in the private network, there is no problem. However, the scenario where a HIP enabled host in the public network initiates a connection to a legacy host in the private network, is a bit more difficult to solve. A quick fix to the problem would be to give the legacy host two addresses; one to be used when contacting hosts in the public network, and one that could be used by hosts wanting to contact this host. The IP address of the legacy host that would be used when initiating a connection would be configured into the HIP proxy, and as a result the connection could benefit from HIP. The other address of the legacy host, used by hosts in the private network when initiating a connection, would not be configured into the proxy. This way the HIP proxy would never interfere with a packet that has that legacy host address as source or destination. Another solution will be presented in Section 6.2.

6.1.3.2 Unexpected problems

During testing we also discovered a problem that is not a HIP specific problem, but rather an IPsec problem. When processing outgoing packets, the SA to be used is located with the help of the source and destination addresses of the packet. When HIP is used, the addresses should actually be HITs instead of IP addresses. If HITs are not used, it is difficult to find the correct outgoing SA if there are multiple SA pairs between two hosts. This IPsec problem should not exist when using HIP, but the current version of the used HIP implementation was not done with a HIP proxy in mind and did not yet have all the features implemented.

The used HIP implementation creates and browses the SAs based on the source and destination IP addresses. Since the IP address of the HIP proxy will be one of the two

addresses connected to each SA it creates, there will likely be problems. We found that if two legacy hosts wish to communicate with the same HIP enabled host in the public network, only one of them will succeed. When the HIP proxy receives packets from the legacy hosts, and is about to send the packets to the HIP enabled host, it finds the SA to be used based on the source and destination addresses of the packet. However, since the SA pairs created for both legacy hosts have the same source and destination addresses, the correct SA will not necessarily be selected. When selecting an SA, the first SA that matches the source and destination addresses will be selected. So all traffic to a specific HIP enabled host will always use the same SA. Thus only the packets from the legacy host, which SA is found first in the SA list, will get through correctly. The other legacy host(s) that wishes to communicate with that same HIP enabled host will not succeed. This problem was solved by modifying the HIP implementation. When the HIP proxy prototype implementation was tested, only the IP addresses were used for finding the correct SA. The feature of using HITs for resolving SAs had not been activated previously, since there had not been a need for it before the HIP proxy was introduced.

A lack in the functionality of the HIP proxy was also discovered; the HIP proxy cannot handle mobile HIP hosts that change their location while they still have connections open via the proxy. When the HIP host moves to a new location it sends a location update message to its communication peers, which might include a HIP proxy. When the proxy machine receives the update message it updates the SA information it has stored for the connections to the HIP host. However, the information of the new location does not reach the HIP proxy application. Consequently no new connections to the mobile HIP host can be established since the proxy can only try to contact the HIP host at its old locator. A solution to this is to update the HIT-IP mapping in the proxy e.g. by modifying the configuration file. As a result, new connections to the new locator of the HIP host can be established. However, this solution introduces a new problem; the proxy will now have two HIT-IP mappings with the same HIT, but every time the proxy needs to find a HIT-IP mapping matching that HIT it will first find the newest mapping. Therefore the connections that were established to the old locator of the HIP host will stop receiving packets from that locator. Instead, they will begin

receiving packets from the new locator of the HIP host without realizing that the packets actually are coming from the same peer host. The legacy hosts connected to the old locator of the HIP host will still be able to send packets to the HIP host, but the replies will be discarded because the source address does not match any of the connections the legacy host has. One way to solve this problem would be to store the old locators of mobile HIP hosts, along with state information for connections, in the HIP proxy. However, this adds to the complexity of the proxy and it is probably not the optimal solution for the problem. This problem needs to be addressed in the future.

6.2 Future work and expectations

The HIP proxy implementation presented in this thesis is a mere first prototype, so there are many ways to improve it. The used implementation environment results in some need for improvement; the target was a HIP proxy in a GGSN in a 3G network, while the implementation actually was done for a computer network. The operation of these two networks differs remarkably. Some features, e.g. configuring the proxy from the command prompt, were dropped because they were thought not to be essential for this work, or found to be too difficult to implement compared to the possible gain. In addition, some areas needing improvement, e.g. enabling the use of HITs for looking up SAs in the HIP implementation, were discovered during testing. There are many ideas on how to improve the implementation; five of the more important ones will now be introduced.

The current simple HIT-to-IP mapping works and is sufficient for this prototype. Since the configuration file is checked at some intervals, and not necessarily when it has been modified, there are some problems; there might be unnecessary checks, which are time-consuming, or then the checks are not done frequently enough, resulting in the lists not being up-to-date. A big improvement would be to have the lists updated dynamically. This is necessary in a real 3G environment.

Another problem related to the proxy configuration concerns the creation of HIP contexts. Currently, all the possible new contexts are created when the configuration file is read, which is note wise when resources are scarce. The optimal solution would be that HIP contexts are created when they are needed, i.e. when a connection is initiated. With only a few nodes configured into the proxy, creating the contexts in advance is not a problem, but with thousands of hosts there can be over a million of connection possibilities, and thus also contexts.

In Section 6.1.1 it was mentioned that SSH was used for testing the HIP proxy. However, using services that provide end-to-end security through the HIP proxy is unnecessary; there is no need for encrypting the traffic twice. The proxy should notice when secured, e.g. ESP protected, traffic arrives from the private network, and just forward those packets. This would also solve the problem mentioned in Section 6.1.3.1 about two HIP enabled hosts communicating via the proxy.

The problem mentioned in Section 6.1.3.1, where a HIP enabled host contacts a legacy host that resides in the private network needs to be solved. One solution, even if probably not the optimal one, would be that the HIP proxy maintains state information of active connections initiated from the public network. By forwarding all packets belonging to these connections, the connections would work. However, this solution has some problems; if a legacy host of one of these connections wants to establish a secured connection to the peer host in the public network using the HIP proxy, the proxy will not function properly. It still believes that the packets are part of the connection that it has a state for, and thus will not initiate the HIP base exchange. Instead it will forward the packets unprotected.

It was earlier noted that HIP mobility does not work flawlessly with the prototype HIP proxy. A legacy host connected to the old locator of the mobile HIP host cannot be informed of the new locator when the HIP host moves. Consequently the legacy host will continue sending packets to the old locator of the HIP host but the replies will be coming from the new locator and therefore be dropped. To solve this problem the proxy could maintain a list of locators used by each of the HIP hosts. The proxy

performs a kind of NAT while processing the received packets; packets going to the public network get the new address of the HIP host as destination address, while packets going to the private network get the old address of the HIP host as the source address. However, this does not solve the problem completely since new connections to the HIP host can be initiated for each address that it uses. As a result the proxy would need to keep some state information for each connection so that it would know which address each legacy host is expecting packets from. This is probably not the optimal solution for the problem and more thought needs to be put on how to solve it efficiently.

Enhancing the HIP proxy with the discussed features will result in a complete solution. The enhanced version would be capable of handling all regular scenarios, and should not have problems with more complex ones either. It could well be used in computer networks for providing security when contacting HIP enabled host from hosts in a private network. The usefulness of a HIP proxy relies heavily on the availability of servers using HIP. Since HIP is still being developed and not yet being widely used, there is currently no big demand for HIP proxies. Once HIP begins spreading to regular Internet users, the demand for a HIP proxy will grow. The HIP proxy might also help HIP to spread more efficiently; using a HIP proxy, end-users do not need to upgrade their IP stack to benefit from HIP. With a HIP proxy, users can test and evaluate the services provided by HIP without having to do any kind of configuration.

When the use of a HIP proxy has been verified in a computer network environment, it might be time for creating a HIP proxy for a 3G network. Based on reasoning presented earlier, the target node for a HIP proxy would be the GGSN. However, when the time comes, some more analyzing will be done and other possibilities will be considered. Still, the services that a HIP proxy can provide are valuable for the 3G network users.

7 Conclusions

Security in digital communication is a big issue today. The focus is mainly on the security of personal computers, but the concern should likewise apply to mobile telephones accessing services in the Internet. We have presented a solution that provides communication security for mobile telephones connected to the Internet.

HIP is a protocol that can provide many useful services to users of 3G networks, the main advantage being security. However, making 3G UEs HIP enabled in large scale is difficult because the users are required to participate in the update process by taking the UE to a designated place for the upgrade. For an adequate level of added security HIP is only needed in the Internet, so the HIP connection may well end at the border between the Internet and the 3G network. Placing a HIP proxy in a GGSN is all that is needed to provide HIP services to 3G network users. It provides the Internet connections with HIP, and does not require anything from 3G network users.

At this stage, while designing and implementing the first prototype, it was not necessary to make the HIP proxy for an actual GGSN. The implementation presented in this thesis is for a computer network environment and is enough for proving the concept of a HIP proxy. With the functionality of the HIP proxy prototype implementation verified in a computer network, the next step would be to actually implement a HIP proxy for a GGSN.

To provide needed services for the HIP proxy we had to modify the used operating system, FreeBSD. Without any previous knowledge of the source code and its organization the start was a bit shaky. As the work went on, the code became more familiar, and one came to grasp the whole picture. The functionality of the actual HIP proxy application is basically not complex. However, as with practically any software project, the first attempt was not 100% successful and modifications and fixes were needed. The development proceeded incrementally, adding new features as the

previous ones were found working and new attributes were thought of and deemed necessary.

Even if the current version of the HIP proxy does perform almost all tasks required by a HIP proxy, be it in a computer network or in a GGSN, there are still issues that need to be solved. Especially a solution for HIP mobility via a HIP proxy needs to be produced. Having a HIP proxy operating with computer networks results in more connection scenarios than what would be the case if the proxy was in a 3G network. Some of these additional scenarios have proven to be problematic and it is doubtful that the optimal solution for them has been reached.

The performance of the proxy in a real environment, with many connection initiations and much traffic, is still unproven. Based on the small-scale testing done with the proxy, it performs well. However, it is expected that method used for managing the HIT-to-IP mappings, as well as updating the mappings, will prove to be inadequate when dealing with a large user base. A more efficient method is required.

Still, with its flaws, of which can be expected from a first prototype, the HIP proxy performs its tasks. Using the proxy, legacy hosts in the private network can establish HIP associations with HIP enabled hosts in the public network via the proxy. The proposed changes to the HIP proxy are mainly aimed at getting the proxy better suited for large-scale networks.

The current demand, if any, for a HIP proxy is probably satisfied with this HIP proxy implementation and its features. However, the proxy needs to be further developed and enhanced to also fill future needs. With HIP widely in use, there will be a demand for a HIP proxy. The fact that the IP protocol used by the mainstream is IPv4 might affect the desirability of this IPv6-only HIP proxy. The proxy can quite easily be converted to an IPv4 HIP proxy, but a real improvement would be to make it IP version independent. Currently HIP is still being developed and the use of HIP is mainly limited to the developers of HIP implementations. However, in the near future HIP can be a security solution to be reckoned with, which could be utilized e.g. in e-commerce.

List of References

[1] "General Packet Radio Service (GPRS); Service description; Stage 2", 3GPP TS 23.060 version 6.3.0, December 2003.

[2] "Universal Mobile Telecommunications System (UMTS); Service aspects; Service principles", 3GPP TS 22.01 version 3.3.0, October 1998.

[3] Postel, "Internet Protocol", RFC 791, September 1981.

[4] Bradner, Mankin, "The Recommendation for the IP Next Generation Protocol", RFC 1752, January 1995.

[5] Deering, Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 1883, December 1995.

[6] Robert Zakon, "Hobbes' Internet Timeline", http://www.zakon.org/robert/internet/timeline/, [Referenced 12. February 2004].

[7] Atkins, Stallings, Zimmermann, "PGP Message Exchange Formats", RFC 1991, August 1996.

[8] Dierks, Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[9] Kent, Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[10] Perkins, "IP Mobility Support", RFC 2002, October 1996.

[11] Moskowitz, Nikander, Jokela, "Host Identity Protocol", Internet draft, work in progress, draft-moskowitz-hip-08, October 2003.

[12] Eriksson, Faloutsos, Krishnamurthy, "PeerNet: Pushing Peer-to-Peer Down the Stack", February 2003.

[13] Clark, Braden, Falk, Pingali, "FARA: Reorganizing the Addressing Architecture", August 2003.

[14] Crawford, Mankin, Narten, Stewart, Zhang, "Separating Identifiers and Locators in Addresses: An Analysis of the GSE Proposal for IPv6", Internet draft, expired, draft-ietf-ipngwg-esd-analysis-05.txt, October 1999.

[15] Stoica, Adkins, Zhuang, Shenker, Surana, "Internet Indirection Infrastructure", August 2002.

[16] Harkins, Carrel, “The Internet Key Exchange (IKE)”, RFC 2409, November 1998.

[17] Rescorla, “Diffie-Hellman Key Agreement Method”, RFC 2631, June 1999.

[18] Anthony Anderberg, “History of the Internet and Web”, http://www.anderbergfamily.net/ant/history/, [Referenced 12 February 2004].

[19] Leonard Kleinrock, “Information Flow in Large Communication Nets”, http://www.lk.cs.ucla.edu/LK/Bib/REPORT/PhD/, May 1961, [Referenced 12 February 2004].

[20] Kalin, “A Simplified NCP Protocol”, RFC 60, July 1970.

[21] Vint Cerf, “A Partial Specification of an International Transmission Protocol”, http://www.cs.utexas.edu/users/kata/HISTORY/Cerf.pdf, [Referenced 12 February 2004].

[22] Postel, “Comments on Internet Protocol and TCP”, IEN 2, August 1977.

[23] Peterson, Davie, “Computer Networks – A Systems Approach” 2nd edition, ISBN 1-55860-577-0, Morgan Kaufmann, 2000.

[24] Plummer, “An Ethernet Address Resolution Protocol”, RFC 826, November 1982.

[25] Jokela, Nikander, Melen, Ylitalo, Wall, “Host Identity Protocol: Achieving IPv4 – IPv6 handovers without tunneling”.

[26] IETF-58 HIP BOF Summary and Minutes, http://www.ietf.org/proceedings/03nov/minutes/hipbof.htm, [Referenced 12 February 2004].

[27] Kent, Atkinson, “IP Encapsulating Security Payload”, RFC 2406, November 1998.

[28] Housley, Ford, Polk, Solo, “Internet X.509 Public Key Infrastructure Certificate and CRL Profile”, RFC 2459, January 1999.

[29] Ellison, Schneier, “Ten Risks of PKI: What You’re Not Being Told About Public Key Infrastructure”, Computer Security Journal – Volume XVI, November 1, 2000.

[30] Kent, Atkinson, “IP Authentication Header”, RFC 2402, November 1998.

[31] Freier, Karlton, Kocher, “The SSL Protocol Version 3.0”, Internet draft, expired, draft-freier-ssl-version3-02.txt, November 1996.

[32] "General description of a GSM Public Land Mobile Network (PLMN)", GSM 01.02 Version 6.0.1: November 1998, ETSI.

[33] The GSM Association, http://www.gsmworld.com/index.shtml, [Referenced 12 February 2004].

[34] Ericsson Telecom AB, Telia AB, Studentlitteratur AB, "Understanding Telecommunication 2", ISBN 91-44-00214-9, 1998.

[35] "Digital cellular telecommunications systems (Phase 2+)", ETSI TS 122 078 version 5.11.0, September 2003.

[36] Cai, Goodman, "General Packet Radio Services in GSM", IEEE Communications Magazine, October 1997.

[37] Status of IMT-2000 Deployments, http://www.itu.int/ITU-D/imt2000/implementation.html, [Referenced 12 February 2004].

[38] Holma, Toskala, "WCDMA for UMTS – Radio Access For Third Generation Mobile Communications", ISBN 0 471 72051 8, Wiley, 2000.

[39] "3G Security; Security Architecture", 3GPP TS 33.102 version 6.0.0, September 2003.

[40] "A Guide to 3rd Generation Security", 3GPP 3G TR 33.900 version 1.2.0, January 2000.

[41] Biryukov, Shamir, Wagner, "Real Time Cryptanalysis of A5/1 on a PC", 2000.

[42] The 3rd Generation Partnership Project (3GPP), http://www.3gpp.org/, [Referenced 12 February 2004].

[43] "3G Security; Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 1: f8 and f9 Specification", 3GPP TS 35.201 version 5.0.0, June 2002.

[44] "3G Security; Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 2: KASUMI Specification", 3GPP TS 35.202 version 5.0.0, June 2002.

[45] Eggert, "Host Identity Protocol (HIP) Rendezvous Mechanisms", Internet draft, work in progress, draft-eggert-hip-rendezvous-00.txt, February 2004.

[46] Nikander, Ylitalo, Wall, "Integrating Security, Mobility and Multi-homing in a HIP Way", February 2003.

[47] Krawczyk, Bellare, Canetti, “HMAC: Keyed-Hashing for message authentication”, RFC 2104, February 1997.

[48] Egevang, Francis, “IP Network Address Translator (NAT)”, RFC 1631, May 1994.

[49] Postel, “Transmission Control Protocol”, RFC 793, September 1981.

[50] man: route, http://www.hmug.org/man/8/route.html, [Referenced 9 July 2004].

[51] HIP for BSD Project, http://hip4inter.net/, [Referenced 8 July 2004].

[52] The FreeBSD Project, http://www.freebsd.org/, [Referenced 8 July 2004].

[53] man: divert, http://www.hmug.org/man/4/divert.html, [Referenced 9 July 2004].

[54] man: socket, http://www.hmug.org/man/2/socket.html, [Referenced 13 July 2004]

[55] FreeBSD/Linux Kernel Cross Reference, http://fxr.watson.org/fxr/find, [Referenced 9 July 2004].

[56] man: ip6fw, http://www.hmug.org/man/8/ip6fw.html, [Referenced 9 July 2004].

[57] Postel, “Internet Control Message Protocol”, RFC 792, September 1981.

[58] Ethereal: A Network Protocol Analyzer, http://www.ethereal.com/, [Referenced 19 July 2004].

[59] man: tcpdump, http://www.hmug.org/man/1/tcpdump.html, [Referenced 19 July 2004].

[60] man: ping6, http://www.hmug.org/man/8/ping6.html, [Referenced 19 July 2004].

[61] SSH Communication Security, http://www.ssh.com/, [Referenced 19 July 2004].