

Mobile Implementations of Virtual Private Networks

Mathias Mårtens

Arcada University of Applied Sciences
Information Technology

Helsingfors 2010

DEGREE THESIS	
Arcada	
Degree Programme:	Information technology
Identification number:	2661
Author:	Mathias Mårtens
Title:	Mobile Implementations of Virtual Private Networks
Supervisor:	Göran Pulkkis
Commissioned by:	Arcada
<p>Abstract:</p> <p>This thesis work examines mobile Virtual Private Networks (VPN) and how they can be implemented. Key network protocols, security protocols and security frameworks that are presently used in mobile computing and VPNs are examined. The network protocols examined are Mobile IP, which has been in commercial use for some time, and Host Identity Protocol (HIP), which is still in its development phase and is not commercially implemented yet. The security frameworks and protocols examined are IPsec, MOBIKE, SSL/TLS and SSH. Both IPsec and TLS/SSL are recognized VPN protocols. SSH is also a common protocol but it is more often used for secure remote login and secure file transfer. The biggest issue with mobile VPN is to make legacy hosts and software mobile. Depending on what network protocol is being used this is done differently. Solutions based on Mobile IP add network hardware. HIP does not demand any additional network hardware. HIP implements a new layer into the network protocol stack making the mobility transparent to higher layer software. Since the HIP is still under development, testing solutions based on HIP has been a difficult task. However, the tests using HIP have been surprisingly successful. The important sources for this thesis work have been documentation provided by IETF as well as my own testing of different combinations of protocol implementation and software.</p>	
Keywords:	Virtual Private Network, Host Identity Protocol, Mobile IP, MOBIKE, TLS/SSL, SSH, IPsec.
Number of pages:	85
Language:	English
Date of acceptance:	22.4.2010

EXAMENSARBETE	
Arcada	
Utbildningsprogram:	Informationsteknik
Identifikationsnummer:	2661
Författare:	Mathias Mårtens
Arbetets namn:	Mobila virtuella privatnät
Handledare:	Göran Pulkkis
Uppdragsgivare:	Arcada
<p>Sammanfattning:</p> <p>I detta examensarbete undersöks mobila virtuella privatnät (VPN) och hur dessa kan realiserars. Centrala protokoll, datasäkerhetsprotokoll och datasäkerhetsramverk, som används i mobil informationshantering idag, undersöks. Nätverksprotokollen som undersöks är Mobile IP, som redan en tid varit kommersiellt realiserad i mjukvaror, samt Host Identity Protocol (HIP), som ännu är i utvecklingskedet och ännu inte har realiserats kommersiellt. Datasäkerhetsprotokollen och ramverken som undersökts är IPsec, MOBIKE, SSL/TLS och SSH. Både IPsec och SSL/TLS är erkända VPN-protokoll. SSH är inte ett protokoll som vanligtvis använts för VPN utan för bl.a. säker distansinloggning och säker överföring av filer. Den största utmaningen med mobil VPN är huruvida existerande klienter kan utnyttja den mobila funktionaliteten utan stora uppdateringar. Hur detta görs beror på olika lösningar, men lösningar som använder samma protokoll är generellt lika. För Mobile IP utökas nätverket med ny hårdvara. HIP kräver inga nätverksändringar, eftersom problemet lösts mjukvarumässigt i ett klientprogram som ändrar protokollstacken. Ändringen möjliggör mobilitet och är osynlig för de övre lagren i protokollstacken. Eftersom HIP ännu är i utvecklingskedet så har testningen varit rätt svår. Den testning som har gjorts har dock varit överraskande framgångsrik. De viktigaste källorna som använts är IETF-dokument och de tester som utförts.</p>	
Nyckelord:	Virtual Private Network, Host Identity Protocol, Mobile IP, MOBIKE, TLS/SSL, SSH, IPsec.
Antal sidor:	85
Språk:	Engelska
Datum för godkännande:	22.4.2010

TABLE OF CONTENT

List of acronyms	6
List of figures.....	10
1 Introduction	12
1.1 Background.....	12
1.2 Aims and goals.....	13
1.3 Structure of the thesis.....	13
2 VPN implementations	14
2.1 IPsec	14
2.1.1 Building blocks of IPsec	14
2.1.2 IPsec in VPN scenarios.....	22
2.2 MOBIKE.....	25
2.2.1 MOBIKE features	25
2.2.2 MOBIKE scenarios	26
2.3 SSL/TLS	27
2.3.1 Building blocks in SSL/TLS.....	28
2.3.2 SSL/TLS in VPN scenarios.....	29
2.4 SSH – Secure Shell	29
2.4.1 Transport Layer Protocol.....	30
2.4.2 Authentication Protocol	30
2.4.3 Connection Protocol.....	30
2.4.4 SSH VPN Scenario	31
3 Mobility protocols	32
3.1 Mobile IP	32
3.1.1 Mobile IPv6 and MEXT	34
3.2 VPN over MIP	35
3.2.1 MIP and IPsec	36
3.2.2 VPN scenario, MIPv4 with two HAS.....	36
3.3 HIP	41
3.3.1 VPN over HIP	44
3.3.2 HIP Proxy	44
3.3.3 Overlay Convergence Architecture for Legacy Applications.....	45
3.3.4 HIP VPN Gateway	47

3.4	Theoretical conclusion	47
4	Evaluation of mobile VPN solutions.....	49
4.1	Birdstep SafeMove	49
4.1.1	Overview	49
4.1.2	Mobility	49
4.1.3	Conclusion	49
4.2	Cisco Mobile VPN	50
4.2.1	Overview	50
4.2.2	Network scenario	50
4.2.3	Cisco Mobile VPN specific configuration example	51
4.2.4	Conclusion	52
4.3	IETF standard for Mobile IPsec VPN based on Mobile IPv4 and MOBIKE	52
4.4	Mobile IPsec VPN based on Mobile IPv6.....	53
5	Practical studies	55
5.1	Test environment and tools	55
5.1.1	VMware Player	55
5.1.2	Ubuntu.....	56
5.1.3	OpenVPN	56
5.1.4	Wireshark	57
5.1.5	InfraHIP project	58
5.2	Mobile VPN with HIP – practical testing	59
5.2.1	InfraHIP and openVPN.....	59
5.2.2	InfraHIP and SSH.....	61
5.2.3	OpenHIP and OpenVPN.....	63
5.2.4	OpenHIP and SSH	68
5.2.5	Ocala – HIP proxy	74
5.3	Practical conclusion.....	74
6	Discussion	75
	List of references	77
	APPENDIX A: MOBIKE - MN CHANGES POINT OF ATTACHMENT	
	APPENDIX B: COMPILATION GUIDE FOR WIRESHARK WITH HIP PATCH	
	APPENDIX C: OPENVPN CONFIGURATION AND SETUP	

LIST OF ACRONYMS

AAA – Authentication, Authorization and Accounting

AES – Advance Encryption Standard

API – Application Programming Interface

ARP – Address Resolution Protocol

CN – Corresponding NODE

CoA – Care of Address

CPU – Central Processing Unit

DES – Data Encryption Standard

DHCP – Dynamic Host Configuration Protocol

DMZ – De-Militarized Zone

DNS – Domain Name Service

DNSSEC – Domain Name System Security Extensions

ESP – IPsec Encapsulating Security Payload

FA – Foreign Agent

GPRS – General Packet Radio Service

GW – Gateway

HA – Home Agent

HI – Host Identifier

HIIT – Helsinki Institute of Information Technology

HIP – Host Identity Protocol

HIPL – Host Identity Protocol for Linux

HIT – Host Identity Tag

HMAC – Hash Message Authentication Code

HTTP – Hypertext Transport Protocol

ICMP – Internet Control Message Protocol

i-CoA – internal Care of Address

ICV – Integrity Check Value

i-HA – internal Home Agent

i-HoA – internal Home Address

IKE – Internet Key Exchange

IKEv2 – Internet Key Exchange Version 2

i-MIP – internal MIP

IP – Internet protocol, referring to either IPv4 or IPv6

IPsec – Internet Protocol Security

IPv4 – Internet Protocol version 4

IPv6 – Internet Protocol version 6

IV – Initialization Vector

LAN – Local Area Network

LSI – Local Scope Identifiers

LTS – Long Time Service

MAC – Message Authentication Code

ME – Mobile Equipment

MIP – Mobile Internet Protocol, referring to either MIPv4 or MIPv6

MN – Mobile Node

MOBIKE – Mobile Internet Key Exchange

MR – Mobile Router

NAT – Network Address Translation

NAT-T – Network Address Translation Traversal

OC – Overlay Convergence

OCALA – Overlay Convergence Architecture for Legacy Applications

OSI – Open System Interconnection

PIN – Personal Identification Number

PKI – Public Key Infrastructure

PPP – Point-to-Point Protocol

RAM – Random Access Memory

RVS – Rendezvous Server

SA – Security Association

SFTP – Secure File Transfer Protocol

SIP – Session Initiation Protocol

SNR – Signal to Noise Ratio

SPD – Security Policy Database

SPI – Security Parameter Index

SSH – Secure Shell

SSL – Secure Sockets Layer

SUM – Secure Universal Mobile

TCP – Transmission Control Protocol

TIA – Tunnel Inner Address

TLS – Transport Layer Security

TTL – Time To Live

UDP – User Datagram Protocol

UMTS – Universal Mobile Telecommunications System

VOIP – Voice Over Internet Protocol

VPN – Virtual Private Network

WLAN – Wireless Local Area Network

x-CoA – external Care of Address

x-HA – external Home Agent

x-HoA – external Home Address

x-MIP – external MIP

LIST OF FIGURES

Figure 1. AH tunnel mode packet (Guide to IPsec VPNs, NIST. 2005).....	15
Figure 2. AH transport mode packet (Guide to IPsec VPNs, NIST. 2005).....	15
Figure 3. The Authentication Header building blocks (Guide to IPsec VPNs, NIST. 2005).....	16
Figure 4. ICMP echo request (PING) with a AH providing for integrity.	17
Figure 5. ESP tunnel mode packet (Guide to IPsec VPNs, NIST. 2005).....	18
Figure 6. ESP transport mode packet (Guide to IPsec VPNs, NIST. 2005).....	18
Figure 7. The ESP packet diagram shows how an ESP packet is constructed and interpreted (RFC 4303, 2005).....	20
Figure 8. Message flow of full handshake.....	29
Figure 9. SSH tunnel used to mount a remote directory. The tunnel is encrypted, and user and server are authenticated.....	31
Figure 10. A traditional network layout	37
Figure 11. The SUM architecture on the same kind of network with Mobile IP as network layer protocol.....	37
Figure 12. Interaction of protocols for SUM using Mobile IP based architecture.	39
Figure 13. Interaction of protocols in SUM architecture with SIP.....	40
Figure 14. Implementation of the SUM architecture with MIP and SIP.....	41
Figure 15. The new Host Identity layer is introduced to the OSI model to decouple the transport layer from the IP layer (InfraHIP Project: Intro, 2009).....	42
Figure 16. HIP Base Exchange from the initial contact to transmission of the first ESP protected message is sent, as proposed (InfraHIP Project: Intro, 2009).....	43
Figure 17. The setup for a HIP proxy between a private network and the Internet.	45
Figure 18. The OCALA architecture, OC layer is placed under the transport layer.	46
Figure 19. An legacy host running a legacy application communicating with a mobile HIP host's legacy application through a OCALA IP-HIP proxy.	47
Figure 20. Cisco Mobile IP VPN configuration Topology Example.	51
Figure 21. Mobile IPsec VPN based on Mobile IPv4 and MOBIKE (R=router).....	52
Figure 22. Computer(A), OpenVPN tunnel successfully initialized. Computer(B) LSI: 1.0.0.50	60
Figure 23. Computer(B), Wireshark log of the OpenVPN tunnel initialization. Computer(A) LSI: 10.0.0.51	60

Figure 24. The successful proof of concept test. In the picture; output from InfraHIP process, output from remote video, explorer showing the remote directory where the video is located.	62
Figure 25. Successful proof of concept, SSH packets addressed with the HIT of Computer(A).....	62
Figure 26. Wireshark dump showing successful update of address of Computer(A) from 192.168.1.44 to 192.168.1.42	63
Figure 27. A point-to-point connection is successfully set up over HIP. Computer(A) is able to login to the server over SSH and the created point-to-point connection.	65
Figure 28. The server, Computer(B), of the test. The figure shows that OpenVPN sets up the connection using the locally configured LSI (1.181.204.51) of computer(A). This is a proof that the connection is set up over HIP.	66
Figure 29. A series of successful updates sent from the client, Computer(A), to the server, Computer(B).	66
Figure 30. A series of successful updates sent from the client, Computer(A), to the server, Computer(B).	67
Figure 31. The client side tries to use the point-to-point interface as its communication link.	67
Figure 32. The HIP Base Exchange is done before SSH negotiation, initiator.....	69
Figure 33. The mounted network drive is ready for use. As seen in the address bar it is using the peers HIT as address.	69
Figure 34. Make-before-break HIP update and the mounted network drive stays operational, (Computer(A)).	70
Figure 35. Make-before-break HIP-update and network drive stays operational, (Computer(A)).	71
Figure 36. HIP update messages being received and processed by the responder (Computer(B)), no re-negotiation needed.	71
Figure 37. Computer(A) HIP update messages are impossible since there is no working interface. Also the mounted network is not operational at this point.	72
Figure 38. The HIP SAis re-negotiated after HIP update messages failed. The mounted network drive is again operational.....	73
Figure 39. Computer(B) refusing HIP update messafes and re-negotiates a HIP SA. ...	73
Figure 40. The success of the different HIP implementations and different programs..	75

1 INTRODUCTION

This B.Sc. thesis addresses issues regarding mobile Virtual Private Network (VPN) technologies and protocols. The main goal of this report is to examine mobility protocols and VPN software and to conduct practical tests with the examined protocols and software. The practical tests will examine different solutions functionality regarding mobility. This B.Sc. thesis contributes to the WISEciti research project on the topic of VPN and Host Identity Protocol. The WISEciti research group include of Helsinki Institute of Information Technology (HIIT), Helsinki University, Arcada University of Applied Sciences and several business partners. For more information see <http://www.cs.helsinki.fi/group/wiseciti/>. The theoretical part has been carried out in an international environment in Kwantlen Polytechnic University in the metropolitan area of Vancouver, Canada. The practical part has also partially been done at Kwantlen. This environment resulted in extra challenges, which resulted in a report with more insight and variety.

1.1 Background

A VPN is a group of computers connected to a private network; a network built and maintained by an enterprise or other institution solely for its own use with limited public-network access. A VPN communicates “securely” over a public network.

The word mobility in this B.Sc. thesis is used to describe not only the physical mobility of a host but also the mobility of the session. A session is a set-up connection between two hosts or nodes, and is preserved while at least one of them topologically moves in the network.

The secure communication channel a VPN sets up and uses is called a VPN session. Traditionally this session is associated with the hosts Internet Protocol (IP) address which makes it highly immobile. To make VPN mobile we can take advantage of mobility protocols that address this precise issue of IP being used as both identifier and locator for communication. Mobile IP (MIP) and Host Identity Protocol (HIP) are the mobility protocols examined in this B.Sc. thesis. They are both great initiatives and strong candidates of becoming widely used. More interesting for this B.Sc. thesis is that they address the issue from different angles.

1.2 Aims and goals

The main goal of this B.Sc. thesis work is to examine different VPN software solutions and different mobility protocols and determine how suited they are for a mobile VPN solution. Also the protocols and VPN technologies main building blocks will be described. The practical part will present mobile VPN solutions using the described software and protocols. There will also be test cases presented for each environment. The test cases should be as similar as possible, in order to draw conclusion from their outcome. The aim for this work is to create a helpful and interesting report regarding VPN and mobility for anyone interested. The practical part will hopefully illustrate different ways to implement VPN in a mobile environment.

1.3 Structure of the thesis

The thesis is divided into two main sections. The first section is theoretical and the second practical. The theoretical section will be divided into sub-sections, the first regarding VPN and the second regarding network protocols. The second part will evaluate different solutions. The evaluations will both be theoretical and practical. The practical evaluation will include test cases and implementations that are closely linked to the theory described in the theoretical part. The theoretical evaluation will review different commercial solutions. For every practical test case a thorough examination and explanation will be done. Conclusions regarding practical functionality of applications and protocols are presented in the practical conclusion section. Pros and cons regarding the software and protocol combinations will also be presented.

Since the focus of the protocol overview is on VPN features of mobile equipment, you might feel that information is missing. All VPN relevant information is included in the thesis, other protocol features are not described. The mobility focus is on node mobility. Network mobility is not a part of this thesis.

2 VPN IMPLEMENTATIONS

There are presently many ways to implement VPNs. Some implementations are more secure and some are easier to use, but the common goal is to create a secure network extension to a Local Area Network (LAN) by tunneling technologies. A VPN tunnel is a secure communication link between two hosts over an unsecure network e.g. the Internet. The link can be of type gateway-to-gateway meaning that it connects two local area networks. A client-to-gateway link connects a single computer to a local area network and a client-to-client link provides a secure channel between two individual computers. The two major and most common ways to accomplish this is using either Internet Protocol Security (IPsec) or Transport Layer Security (TLS) protocol. Whichever protocol is used the same level of security and data integrity will be achieved. The only difference is how this is implemented by the software utilizing the features of the different protocols.

2.1 IPsec

IPsec is a framework of open standards for ensuring private communication over IP networks. IPsec has become the most commonly used network layer security protocol and has reached the status of being a de facto standard in today's information society. IPsec is mainly defined by IETF in RFC 4301 and RFC 4309. There are however many other RFC documents on IPsec since IPsec uses other standards. These standards are defined in their own RFC documents, for example Authentication Header is defined in RFC 4302 and Internet Key Exchange version 2 in RFC 4306. IPsec provides many different types of protection. It maintains confidentiality and integrity, authenticates the origin of data, prevents packet replay and traffic analysis, and provides access protection (Guide to IPsec VPNs, NIST. 2005. RFC 4301, 2005).

2.1.1 Building blocks of IPsec

IPsec has three large building blocks that provide desired features. The blocks also work independently but together they provide confidentiality, integrity, and authentication. Independently they only provide their own feature. The blocks are Authentication Header (AH), Encapsulation Security Payload (ESP), and Internet Key Exchange

protocols (IKE). In this section all these will be described and for each a small example will be provided (Guide to IPsec VPNs, NIST. 2005. RFC 4301, 2005).

Authentication Header

AH provides integrity for the IP header and payload. AH also authenticates the user. AH cannot provide any encryption of the payload, this is done by ESP. Furthermore today's version of ESP has as well as AH the ability to authenticate data. One might therefore think that AH is obsolete. The difference is that AH can provide authentication for the whole IP packet both header and payload, but ESP can only authenticate the payload. However there is IPsec software that solely relies on ESP for authentication today, but a lot of legacy software still uses AH (Guide to IPsec VPNs, NIST. 2005. RFC 4301, 2005. RFC 4302, 2005).

AH have two different modes, transport and tunnel mode. The difference between these two is that in tunnel mode AH creates a new IP header for the whole IP packet, and therefore the term tunnel is used. Transport mode AH does not do this. If the tunnel end-point is a gateway then the final destination is a private network behind the gateway. Therefore packets have to be addressed to the gateway and a new IP header is attached since the final destination must not be changed. If the original IP header is changed then the checksums cannot be verified. Transport mode is mainly used in end-to-end architectures. AH is used to protect the integrity of the whole IP packet. Figure 1 and Figure 2 illustrate the AH packet in tunnel and transport mode.

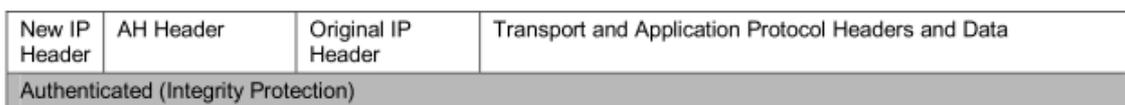


Figure 1. AH tunnel mode packet (Guide to IPsec VPNs, NIST. 2005).

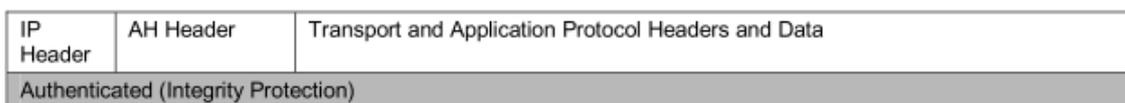


Figure 2. AH transport mode packet (Guide to IPsec VPNs, NIST. 2005).

AH protects the integrity of an IP packet by calculating a checksum of the packet. The checksum is used for verification by the receiver. There are in the IP header some fields that change during transport, for example Time To Live (TTL), and the IP header

checksum. These fields are not included in the AH integrity calculations. These issues also explain why AH often is incompatible with Network Address Translation (NAT). The IP source and destination fields are included in the AH integrity calculations and should not be changed. If a NAT device changes the IP destination from a private to a public address then the AH integrity calculation done by the recipient would not match the checksum passed along with the packet (Guide to IPsec VPNs, NIST. 2005. RFC 4301, 2005. RFC 4302, 2005).

The integrity provided by AH is based on a Message Authentication Code (MAC) which is encoded with a keyed hash algorithm. The produced hash is added to the packet and the packet is sent to the recipient. The keyed hash algorithm produces a hash of both the message and the secret key. The key is shared by both the sender and the recipient. Upon the arrival of the packet the recipient regenerates the hash by using the same shared key and the message from the packet. If the hashes are equal, then the integrity of the message is confirmed. AH uses a Hash Message Authentication Code (HMAC) with two key hashes. Examples of keyed hash algorithms for point-to-point are HMAC-MD5 and HMAC. Another common MAC algorithm is Advanced Encryption Standard (AES) Cipher Block Chaining MAC (AES-XCBC-MAC96). There are many different strategies being developed for multicast connections (Guide to IPsec VPNs, NIST. 2005. RFC 4301, 2005. RFC 4302, 2005).

The AH header consists of six different fields, next header, payload length, reserved, security parameters index, sequence number and authentication information. Some of these are self-explanatory and some are more complex. The Security Parameters Index (SPI) is a unique identifier for the connection. It is used by the recipient in combination with destination IP and IPsec protocol type to determine which Security Association (SA) is being used. Figure 3 depicts the different fields found in a AH header.

Next Header	Payload Length	Reserved
Security Parameters Index		
Sequence Number		
Authentication Information		

Figure 3. The Authentication Header building blocks (Guide to IPsec VPNs, NIST. 2005).

A connection consists of two one way connections. The sender and the receiver have their own SPI values corresponding to their connections. Each packet is assigned an incrementing sequence number, this provides defense against replay attacks and denial of service attacks. The actual HMAC value, which is used for integrity checking by the recipient, is located in the authentication information (Guide to IPsec VPNs, NIST. 2005. RFC 4301, 2005. RFC 4302, 2005).

Figure 4 shows the whole packet if a AH were to be added to a PING request (Guide to IPsec VPNs, NIST. 2005).

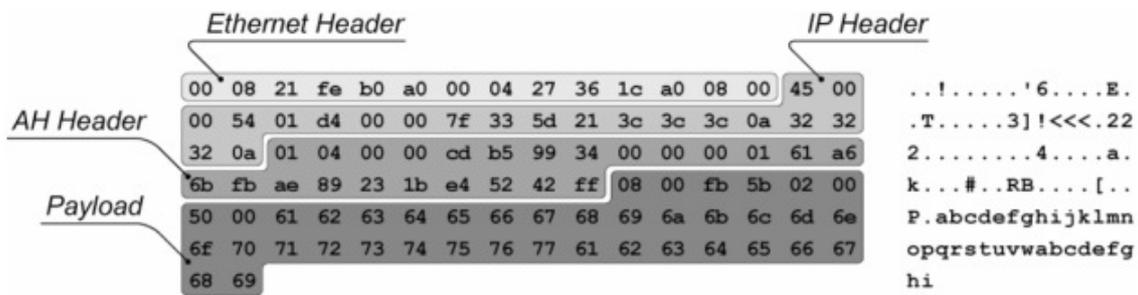


Figure 4. ICMP echo request (PING) with a AH providing for integrity.

Encapsulation Security Payload

ESP provides for integrity and data origin authentication which together are called integrity. Because ESP is connectionless it is provided on a packet-to-packet basis. Integrity calculations are done for each packet. Therefore the data origin is authenticated. The original ESP only provided confidentiality by encrypting the payload. ESP still provides this feature alongside the added integrity. ESP can provide integrity and confidentiality separately and therefore there is three different service combinations; confidentiality only, integrity only, and confidentiality and integrity (RFC 4301, 2005. RFC 4303, 2005)

ESP has two different modes. It can operate in tunnel mode and in transport mode. In tunnel mode a new IP header is attached, which carries a destination address, usually a security gateway. The inner IP header stays unchanged and is the ultimate destination for the packet. ESP allows mixed inner and outer IP versions i.e. IPv6 over IPv4 or IPv4 over IPv6. In tunnel mode the whole inner IP packet is secured by ESP. Figure 5 depicts

an ESP packet in tunnel mode.

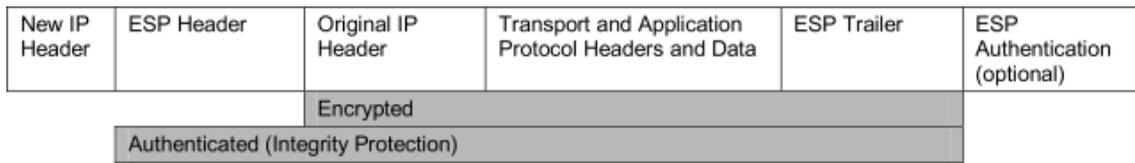


Figure 5. ESP tunnel mode packet (Guide to IPsec VPNs, NIST. 2005).

In transport mode the ESP is inserted after the IP header, leaving it unprotected. When combining ESP and AH in IPv4, the ESP will be placed after the IP header and before any next level protocol. AH, if present, will then apply to the ESP header, to the payload, to the ESP trailer, and to the Integrity Check Value (ICV). In Figure 6 is an ESP packet in transport mode illustrated.

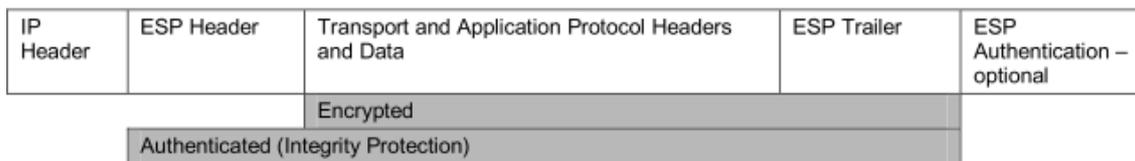


Figure 6. ESP transport mode packet (Guide to IPsec VPNs, NIST. 2005).

If IPv6 is used the ESP header is placed after the hop-by-hop, routing, and fragmentation extension headers. Due to the IPv6 protocol features, destination options and extension headers might be inserted before, after or both before and after the ESP header. It is preferable to have them after the ESP header, since ESP only protects fields found after the ESP header (RFC 4303, 2005).

The ESP applies symmetric cryptography to provide the encryption demanded by IPsec. Both the sender and the receiver use the same shared secret key to encrypt and decrypt the messages. The algorithms being used to encrypt data are block cipher algorithms. Such algorithms divide the data that is to be encrypted into smaller blocks. These blocks are then encrypted in many rounds using the key. Every ESP header contains a SA field that among other things contains the specified algorithm. This is important, since packets might arrive in different order or be dropped. For decryption the same key is used and the process is repeated but in reverse direction. Examples of block cipher algorithms used by ESP are AES Code Block Chaining (AES-CBC), AES

Counter Mode (AES-CTR), and Triple Data Encryption Standard (DES) (3DES) (RFC 4303, 2005. Guide to IPsec VPNs, NIST. 2005).

An ESP consists of the ESP header, the payload, and the ESP trailer. The Header consists of two fields, a SPI field and a sequence number. Both are used the same way as in the AH.

The payload part of the packet contains the encrypted data and its Initialization Vector (IV). The IV is different for each packet and is used for encryption. Therefore, encrypting the same data will result in different payloads. This is an essential feature to protect against cryptanalysis.

The ESP trailer consists of three fields, padding, padding length and next header. Padding is used for three reasons. First use is to make the encrypted data an integral multiple of the block size. Second use is to make sure that the ESP trailer ends on a multiple of 4 bytes. Last use is to conceal the actual length of the payload. The Padding length is a mandatory field holding the size of the padding field. Next header is used in the same way as in AH. It is a value telling the type of the next heading.

Figure 7 depicts the ESP packet diagram and how it is constructed of different parts.

- SPI identifies the security parameters in combination with IP address.
- Sequence number. A monotonically increasing number, used to prevent replay attacks.
- Payload data. The data to be transferred.
- Padding. Used with some block ciphers to pad the data to the full length of a block.
- Pad size. Size of padding in bytes.
- Next header. Identifies the protocol of the payload data. The value of this field is chosen from the set of IP Protocol Numbers defined in the most recent “Assigned Numbers” RFC from the Internet Assigned Numbers Authority.
- Authentication data. Contains the data used to authenticate the packet.

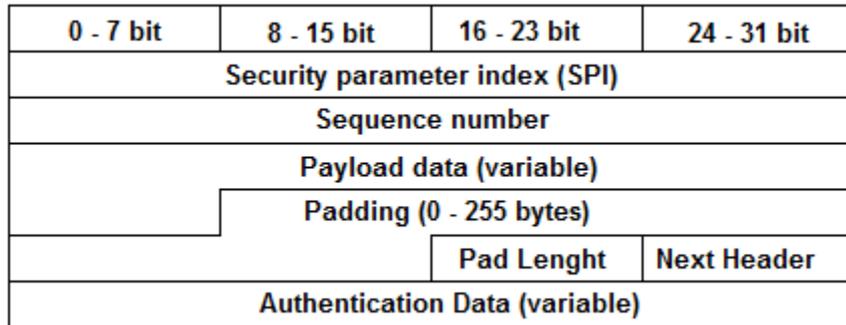


Figure 7. The ESP packet diagram shows how an ESP packet is constructed and interpreted (RFC 4303, 2005).

Internet Key Exchange

The purpose of IPsec is to provide confidentiality, data integrity, access control, and data source authentication. These services are provided by maintaining a shared state, a security association, between the communication peers. This state defines, among other things, the specific services provided to the datagram, which cryptographic algorithms will be used to provide the services, and the keys used as input to the cryptographic algorithms. To establish this shared state in a manual fashion is not realistic. Internet Key Exchange Version 2 (IKEv2) does this dynamically, IKEv2 replaces the three documents, RFCs 2407, 2408, and 2409, which define IKEv1 (RFC 4306, 2005).

The goal of IKE is to set up a shared state, a security association or simply SA. IKE implements mutual authentication for the two communication hosts. When the hosts are authenticated, a SA can be established. The content of a SA is shared secret information that can be used for efficient implementation of ESP and/or AH. The SA also holds a set of cryptographic algorithms that enables the SA to secure its information content (RFC 4306, 2005).

Internet Key Exchange version 1

This will only be a brief presentation of IKEv1 since it is deprecated by the newer IKEv2. IKEv1 is however still being used in legacy software since software older than 2005 is still being used. For more information regarding the IKEv1 refer to RFC 2409 and the Guide to IPsec VPNs (NIST, 2005).

IKEv1 exchanges are defined in two phases, IKE phase 1 that creates an IKE SA and IKE phase 2 that creates an IPsec SA in the channel protected by the IKE SA (Guide to IPsec VPNs, NIST. 2005).

IKE phase 1 has two modes, main mode and aggressive mode. Main mode negotiates the establishment of IKE SA with three pairs of messages. Aggressive mode uses only three messages and is therefore not as reliable but faster. The IKE SA created is bidirectional, providing protection for both sides (Guide to IPsec VPNs, NIST. 2005).

IKE phase 2 has only one mode, quick mode, which uses three messages to establish an IPsec SA. The IPsec SAs created are unidirectional meaning that one SA is needed in each direction. All communication done in phase 2 is protected by an IKE SA (Guide to IPsec VPNs, NIST. 2005).

Internet Key Exchange version 2

IKEv2 works with messaging between the peers. It is always on the requester's responsibility to ensure reliability. The message flow is also always initiated by the initiator by sending a request which is followed by a response from the responder. If there is no response received within a time interval, the initiator must resend its message or terminate the communication (RFC 4306, 2005).

First request/response message pairs between the communicating peers are IKE_SA_INIT and IKE_AUTH. IKE_SA_INIT negotiates cryptographic algorithms, exchanges cryptographic nonces and executes a Diffie-Hellman key-exchange. An IKE_AUTH exchanges identities and certificates. It also establishes the first CHILD_SA. These messages are partly encrypted and integrity protected to hide the identities of communicating hosts and to authenticate all other data fields in the exchange. The encryption keys used are set up by the previous IKE_SA_INIT messages. IKE_AUTH also authenticates the IKE_SA_INIT payload (RFC 4306, 2005).

The exchange could look like this (RFC 4306, 2005):

1. The initiator sends four packets containing different data. The first packet contains its header containing security parameter indexes, version numbers, and various flags. The second packet contains its security association initiator data

that holds information of which algorithms the initiator supports for the IKE_SA. The third packet contains Diffie-Hellman values, and the last packet contains a nonce.

2. The responder chooses one of the suggested algorithm suites presented by the initiator for IKE_SA, completes the Diffie-Hellman exchange and sends a nonce.

After these two steps both the initiator and the responder can generate a SKEYSEED, from which all keys for the IKE_SA will be derived. The keys will be used for encryption and integrity protection. There are also other keys derived from the SKEYSEED, for example keys for creating CHILD_SAs (RFC 4306, 2005).

When all keys are derived and set up, the initiator sends an encrypted and integrity protected packet containing identification, certificates and certificate requests, security association request and traffic selectors. The responder answers with its identity identification, certificates if they have been requested, and completes the CHILD_SA negotiation (RFC 4306, 2005).

2.1.2 IPsec in VPN scenarios

IPsec can be used in many different ways to provide the protection required by different implementations. For VPN IPsec is used to provide authentication, encryption and integrity. In both scenarios the communication is set up by IKE and protected by IKE_SA, when an IPsec tunnel is set up for the VPN traffic. Most common setups for VPN are gateway-to-gateway and host-to-gateway. Gateway-to-gateway provides a secure IPsec communication link between networks. Host-to-gateway uses IPsec client software on the host. The connection is set up between the host and a gateway to provide a secure IPsec connection from any connection point in the Internet.

Gateway-to-gateway

This configuration is used to set up a secure IPsec communication link for a VPN between gateway A and gateway B. User A is inside network A where also gateway A is located. User B is inside network B and gateway B is also located there.

The communication flow, as described by (Guide to IPsec VPNs, NIST. 2005):

1. User A sends regular packets to user B.
2. Network A routes packets to gateway A
3. Gateway A alternates the source IP of the packet by NAT.
4. Gateway A matches the packets to its Security Policy Database (SPD) and determines that they are to be protected by ESP and also determines the address of the destination gateway. If SPD has no pointer to any IKE SA the gateway knows that there is no current IKE SA to protect the particular traffic.
5. Gateway A initiates with gateway B an IKE SA negotiation that will result in an IKE SA.
6. Gateway A wants to establish an ESP connection with gateway B. Gateway A uses parameters set in IKE SA to initiate an IPsec SA negotiation. All communication is protected by the IKE SA. The gateways will use ESP tunnel mode that will provide encryption and integrity. The result of the negotiation is two unidirectional IPsec SAs, one in direction A to B and one in direction B to A.
7. Upon completion of the IPsec SA setup gateway A can finish processing the packets from user A. The packets will be processed according to the SA parameters. This includes adding an IP header to the packets with gateway A as source IP and gateway B as destination, encrypting the data and adding the authentication parameters. Then the packets are sent from gateway A to gateway B.
8. Gateway B receives the packets and uses the unencrypted SPI values to determine which SA to be used. Gateway B processes the packets according to the right SA and validates the content. When the additional IP header has been removed, the integrity has been checked, and the payload has been decrypted, a packet will be sent to the original destination address which is user B.

If user B wants to reply and the IPsec SA is maintained then steps 7 and 8 are repeated for user B. If an IPsec SA is not maintained then the whole process is repeated.

Host-to-gateway

The scenario is the same as the gateway-to-gateway scenario: user A wants to send packets to user B in a different network. The difference is that user A is in a network without a VPN gateway, for example a public network. User B is inside a corporate network and a gateway B provides a VPN connection point. User A must now instead of having a gateway to establish and negotiate the IPsec SA have IPsec software running locally.

1. User A sends regular packets to user B.
2. IPsec of user A matches the packets to its SPD and determines that they are to be protected by ESP and also determines the address of the destination gateway. In SPD there is no pointer to any IKE SA. The IPsec software therefore knows that there is no current IKE SA to protect the particular traffic.
3. IPsec software of user A initiates with gateway B an IKE SA negotiation that will result in an IKE SA.
4. User A IPsec software wants to establish an ESP connection with gateway B. User A's IPsec software uses parameters set in IKE SA to initiate IPsec SA negotiations. All communication is protected by the IKE SA. The ESP tunnel mode will be used to provide encryption and integrity. The result of the negotiation is two unidirectional IPsec SAs one in direction A to B and one in direction B to A.
5. Upon completion of the IPsec SA setup IPsec software of user A can finish processing the packets from user A. The packets will be processed according to the SA parameters. This includes adding an IP header to the packet with IPsec software of user A as source IP and gateway B as destination, encrypting the data, and adding the authentication parameters. Then the packet is sent from IPsec software of user A to gateway B.

6. Gateway B receives the packets and uses the unencrypted SPI values to determine which SA to be used. Gateway B processes the packets according to the correct SA and validates the content. When the additional IP header has been removed, the integrity has been checked, and the payload has been decrypted, the packet will be sent to the original destination address which is user B.

If user B wants to reply and the IPsec SA is maintained, then steps 5 and 6 are repeated but according to suite user B. If an IPsec SA is not maintained, then the whole process is repeated.

2.2 MOBIKE

Mobile Internet Key Exchange (MOBIKE) is an extension of IKEv2 developed to add Mobility and Multihoming features to IKEv2. MOBIKE's goal is to enable efficient management of IPsec and IKE security associations when a host is attached to the Internet by multiple interfaces or if a host changes its point of attachment to the Internet. Presently IKEv2 must negotiate its SA's after any change of the network topology or interface. This is a process demanding a lot of calculations and can also include user interaction, for example asking the user to insert a Personal Identification Number (PIN) for a security token (RFC 4621, 2006).

The problem with IKEv2 is that it assumes that the SA created in the execution of the protocol is between IP addresses of the peers. This means that only one IP address in each host can be used with the designated SA and IKEv2 protocol session. IPsec does not have any documentation describing, suggesting or allowing these pairs to be changed, elsewhere than in Network Address Translation Traversal (NAT-T) (RFC 4621, 2006).

2.2.1 MOBIKE features

MOBIKE addresses this problem by suggesting itself to work on top of the IKEv2. All development in MOBIKE has to be done according to both IKEv2 (RFC 4306) and Security Architecture for the Internet Protocol (RFC 4301). This way the security risks are minimized since they have already been taken in consideration the mentioned RFC documentation. MOBIKE does not try to be a full mobility protocol, provides no

rendezvous mechanism for simultaneous movement, and has no support for route optimization. These features are left out to other levels of the Open System Interconnection (OSI) - model and to other protocols like Mobile IP. MOBIKE focuses solely on what two peers need in order to agree at IKEv2 level and what is required for interoperability. MOBIKE defines new message formats and processing of messages with the defined format. MOBIKE focuses on tunnel mode. Therefore, any application running inside a MOBIKE controlled IPsec tunnel might not even detect any movement since their IP remains the same. Only the headers of the tunnel IP addresses are changed according to any movement of a mobile node (MN). MOBIKE is suggested to be able to perform the following operations;

- Inform the other peer about the peer address set
- Inform the other peer about the preferred address
- Test connectivity along a path and thereby detect an situation with no connection
- Change the preferred address
- Change the peer address set
- Ability to deal with NAT devices

All features are not supported in the current protocol (RFC 4621, 2006).

2.2.2 MOBIKE scenarios

MOBIKE functionality is described through three different scenarios;

- Scenario one; break-before-make. This scenario addresses the problem with a peer already connected to an IPsec gateway and losing the connection and after being unreachable connects through a new point of attachment, and there with a new IP address. MOBIKE's goal here is to continue to use the same SA that was used by the old point of attachment and prevent expensive renegotiation of the SA.
- Scenario two; multihomed. This scenario deals with the issue of having more than one network interface for security, cost or some other reason. This scenario

is not for mobility use but is meant to preserve a connection if anything happens to the primary interface.

- Scenario three; multihomed laptop. This scenario deals with a mobile node (ex: a laptop) having many different ways of staying connected, through wire, Wireless Local Area Network (WLAN), Bluetooth, or Universal Mobile Telecommunications System (UMTS) or General Packet Radio Service (GPRS). All these are always available and depending on cost, throughput, and network availability are used when appropriate. MOBIKE does not consider in what way the preferred connection is determined. The issue is to keep the same SA for IP address changes and between different interfaces. Only the tunnel IP address in the IP packet header is changed so applications running inside the IPsec tunnel will stay unaffected and unknowing.

All scenarios are basically the same. MOBIKE provides for the IPsec and IKEv2 SA's to be changed and they don't have to be renegotiated. For an illustration of how this is done see Appendix A (RFC 4621, 2006. RFC 4555, 2006).

MOBIKE is an important part of making the commonly used IPsec more suitable for mobile implementations. MOBIKE and IPsec working on a mobile protocol like MIP provides a good platform for mobile VPNs.

2.3 SSL/TLS

Secure Socket Layer (SSL) developed by Netscape and its successor TLS developed by IETF are cryptographic protocols providing security and data integrity. SSL/TLS is used to secure traffic over unsecure networks such as the Internet. SSL was first developed by Netscape in the nineties. In the end of the nineties IETF started a workgroup which tried to collect all individual SSL versions and features and make one protocol out of them, TLS. SSL/TLS is a well trusted, spread and used protocol for securing data over the Internet, presently used for e-mail, web browsing, instant messaging, Voice-Over-IP (VOIP), and VPN (G.Magnini, 2005. RFC 5246, 2008).

2.3.1 Building blocks of SSL/TLS

SSL/TLS provides security and data integrity with two sub-protocols, the record protocol and the handshake protocol.

Record Protocol

Record protocol is a layered protocol that is used to encrypt and transmit data. The data is fragmented into blocks that are either left at their original size or optionally they can be compressed. The record protocol applies a MAC to the data, encrypts and transmits the result. The key used to encrypt data is negotiated by the handshake protocol. For a receiver the data is processed in the reverse direction. (RFC 5246, 2008).

Handshake protocol

The handshake protocol is responsible for negotiating a session. To create a session the handshake protocol starts with exchanging hello messages to agree on algorithms, exchange random values, and check for session resumption. After this is done, and no session is resumed, the necessary cryptographic parameters needed for the client and server to agree on a premaster secret are exchanged. Moreover, the server and the client authenticate themselves by exchanging certificates. After authentication and agreement on a secret the actual master secret is generated from the premaster secret and the exchanged random nonces. After this, the handshake protocol terminates by passing the security parameters to the record protocol which uses them to verify that they have been negotiated between the server and the client without tampering (RFC 5246, 2008).

There are also other sub-protocols in use by SSL/TLS such as the alert protocol and the change cipher suite protocol. These are used for error and alert messaging and for changing cipher suite if necessary. A full handshake message flow is illustrated in Figure8.

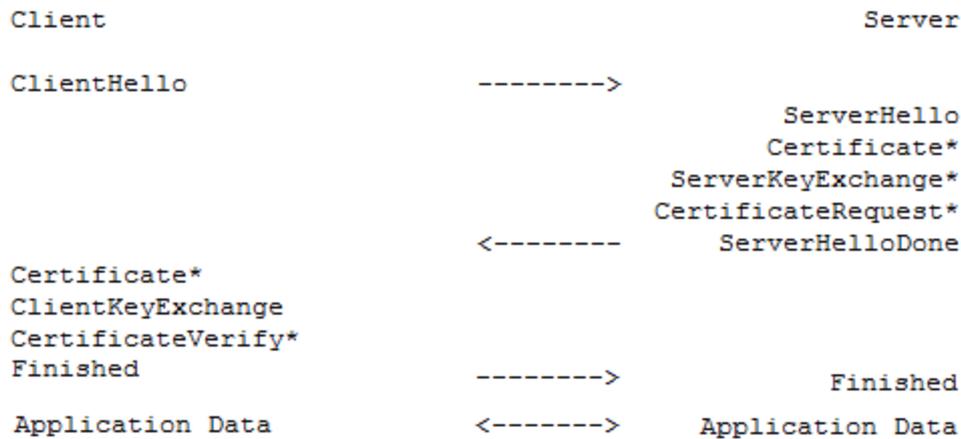


Figure 8. Message flow of a full handshake.

2.3.2 SSL/TLS in VPN scenarios

When a SSL/TLS connection is negotiated, setup correctly, and verified, a tunnel is created from the client to the server. The tunnel provides security and data integrity. The tunnel created is most commonly known for securing Hypertext Transport Protocol (HTTP) traffic but can also be used for VPN. There are a few different VPN solutions based on SSL/TLS one which is OpenVPN. SSL/TLS based VPN software provides the same kind of security as IPsec VPN software provides, a key exchange similar to Diffie-Hellman and encapsulation of the traffic, similar to ESP. There is no standard on how to implement SSL/TLS VPNs, OpenVPN will be presented more in detail in the practical part of the thesis work. All open source SSL/TLS VPN solutions have some common features; easy setup since they run in user space and not in kernel space, well tested security library since they utilize OpenSSL libraries, and not expensive and easy to manage because of open source.

2.4 SSH – Secure Shell

SSH is a network protocol that allows two network interfaces to connect to each other securely. SSH is widely used, the typical use of SSH is secure login to remote computers, secure file transferee (SFTP) and other security sensitive services. The architecture of SSH is defined in RFC 4251, 2006, but the earliest drafts dates back to mid-nineties.

SSH consists of three major protocols, the Transport Layer Protocol, the User Authentication Protocol and the Connection Protocol (RFC 4251, 2006).

2.4.1 Transport Layer Protocol

Transport Layer Protocol, provides for several features essential for the needed security: server authentication, confidentiality, and integrity. When a SSH connection is initiated the Transport Layer Protocol is the first to be used. The Transport Layer Link uses for example a TCP/IP connection, to set up a link that the Authentication Protocol will use (RFC 4251, 2006).

2.4.2 Authentication Protocol

Authentication Protocol, authenticates the client to the server, this is done over the Transport Layer Link. The authenticity of a client can be proven for example with a password, or a key exchange. When the client is authenticated the Connection Protocol will run over it (RFC 4251, 2006).

2.4.3 Connection Protocol

Connection Protocol multiplexes the encrypted tunnel to several logical channels. These logical channels are the actual tunnels which can be used by the user through different application. Common applications using the tunnels are SSH tunneling, SFTP, X11 forwarding and standard interactive secure shells.

SSH has no built in mobile ability. If a SSH connection is to become mobile it has to be set up on a mobile connection. Because the simple usage and flexibility of SSH, setting up an SSH tunnel on a mobile connection is not too difficult. Furthermore, most mobility protocol implementations provide a way for legacy software to communicate through the mobile channel either in IPv4 or IPv6 mode. Since SSH supports both IPv4 and IPv6 this is not a problem (RFC 4251, 2006).

2.4.4 SSH VPN Scenario

Since SSH can authenticate both server and user as well as encrypt the data link it can be used for VPN. Using SSH for VPN can be done by using a tunnel secured with SSH to mount a remote directory. There are different software components that deal with the issues regarding the mounting and how to make the mounted directory to show up as a real hard disk drive. However, they all have the same SSH carrier in common. Figure 9 depicts a simple version of how a SSH tunnel is set up and used with SFTP to mount a remote directory as a local drive.

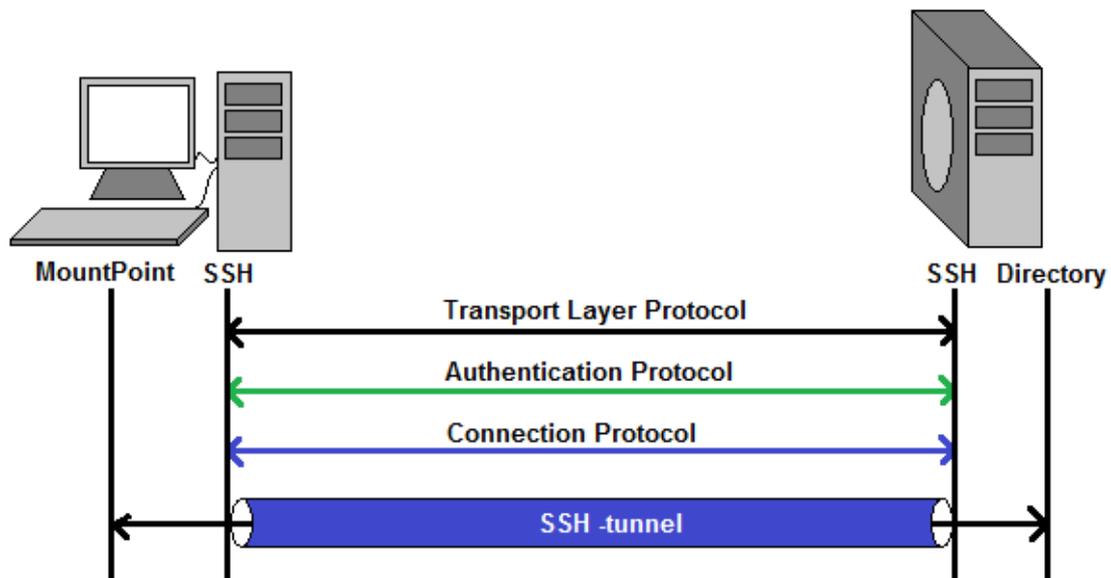


Figure 9. SSH tunnel used to mount a remote directory. The tunnel is encrypted, and user and server are authenticated.

SSH is a widely used and open standard. That's why there are plenty of open source and commercial software implementations SSH and SFTP to mount remote directories. One of these implementations is Expand Drive for Windows which is commercial. There are also GVFS and SSHFS for Linux. (GNOME 2.22 Release Notes, 2008. SSH Filesystem)

3 MOBILITY PROTOCOLS

There are many network protocols that can provide mobility. Two such protocols are MIP and HIP. These two are different in their architecture and work in different ways to provide mobility, multihoming, seamless roaming, and session persistence to the end user. MIP is based on the IP protocol and MIPv6 is the newest MIP protocol. HIP is still a newer protocol and therefore security measures have been incorporated as an essential part of the protocol throughout the development. Mobile IP is today much more commonly used than HIP since both MIPv4 and MIPv6 are further developed. There is also much commercial software using the MIP protocol. HIP is still in the development phase but it has a lot of potential as a mobile protocol. Both protocols are still under development, since there is a continuous evolution in mobile communication and in expectations from mobile services.

3.1 Mobile IP

MIP is a proposed standard based on IP. MIP makes mobility transparent to applications and higher level protocols like Transmission Control Protocol (TCP) (C.E. Perkins 1998)

The MIP standard for IPv4 was first defined in RFC 2002 in 1996 and is therefore the oldest of the protocols regarding mobility examined in this thesis work. Despite the age it has little advantage over HIP regarding mobility, since both protocols are in a seamlessly never ending development stage. MIP is developed to give users the comfort of being mobile, to change their topological location in the Internet and to be connected to a wired, wireless or a cellular network which is public or private. MIP also tries to be transparent to IP so that legacy software and higher protocols don't have to be changed. The technology MIP is using to provide mobility also introduces new terms and new functionality. MN, HA, Foreign Agent (FA) and Care-of Address (CoA), are basic concepts of MIP.

A MN is a host or a router that changes its topological location in the Internet without changing its IP address. The MN can continue to communicate with any other node connected to the Internet with its constant IP address (C.E Perkins, 2002).

A HA is a router in the MN's home network. A HA forwards datagrams to MNs and maintains information about the current location of each MN (C.E Perkins, 2002).

A FA is a router in a MN's visited network. A FA cooperates with the Home Agent to complete the delivery of the datagram to the MN. FAs as routers are not needed in MIPv6, but FA is still a key element in understanding the concept of MIP (C.E Perkins, 2002).

A HA is an IP address that the MN is assigned for a longer period of time. This IP stays unchanged regardless of where in the Internet the MN is attached (C.E Perkins, 2002).

CoA, an IP address reflecting the MNs current point of attachment, is the termination point of a tunnel towards the MN. There are two types of CoA: one is the address of the FA the MN is registered to, and the other type is the address the MN has obtained for one of his network interfaces in the foreign network (C.E Perkins, 2002).

The basics setup and functionality are the same for both the IPv4 and IPv6 versions of MIP. MIP allows the MN to have two IP addresses, the static home address used to identify TCP connections, and the CoA which is the actual topological location. To separate these two in the described way gives the MN a constant address that it can be reached on by any other node in the Internet. Upon the scenario that the MN is in a foreign network the HA will have a record of MN's actual attachment point CoA. This record is continuously updated by the MN when it moves between different locations in the Internet. The MN always appears to be available for traffic since the HA receives all packets that are addressed to the MN's home address. For the MN to be able to receive packets addressed to its interface associated with its HA, the HA must redirect or tunnel the packet to the MN's CoA. At the termination point of the tunnel the packets will be unpacked so that the MN will receive the packets using its interface associated with its static HA. When the MN answers to a node which is communicating with the MN through the HA - CoA tunnel, the MN simply sends its packets directly to the other node and with its interface associated with its HA as sender. This type of asymmetric routing is called triangle routing. The Corresponding Node (CN) will talk to the MN's home address not knowing if the MN is actually in its home network, in a foreign network or if it is moving between different networks. Triangle routing can in some cases be exceptionally inefficient and add unnecessary delay. To avoid this, route

optimization extension is suggested to the basic MIPv4 and is already implemented in MIPv6 (C.E Perkins, 2002.).

3.1.1 Mobile IPv6 and MEXT

To be able to take advantage of new features such as multihoming and network mobility the switch to MIPv6 is inevitable. MIPv6 has its own workgroup under IETF and the description of the group states “Mobile IPv6 specifies routing support which permits an IPv6 host to continue using its home address as it moves around the Internet, enabling continuity of sessions. Mobile IPv6 supports transparency above the IP layer, including maintenance of active transport level sessions. In addition, network mobility (NEMO) mechanisms built on top of Mobile IPv6 allow managing the mobility of an entire network, as it changes its point of attachment to the Internet. The base specifications consist of: RFC 3775 (Mobile IPv6), RFC 3963 (NEMO), RFC 4877 (Mobile IPv6 Operation with IKEv2)”.

The main differences between MIP4 and MIPv6 which are important in this B.Sc. are (RFC 3775. 2004. draft-ietf-mext-rfc3775bis-03.txt. 2009):

- There is no need to deploy special routers as FAs, as in MIPv4. MIPv6 operates in any location without any special support required from the local router.
- MIPv4 route optimization has to be done by a nonstandard set of extensions but MIPv6 has support for route optimization as a fundamental part of the protocol. MIPv6 route optimization can operate securely even without pre-arranged SAs. It is expected that route optimization can be deployed on a global scale between all MNs and correspondent nodes. Support is also integrated into MIPv6 for allowing route optimization to coexist efficiently with routers that perform "ingress filtering”.
- Most packets sent to a MN while away from home in MIPv6 are sent using an IPv6 routing header rather than IP encapsulation, reducing the amount of resulting overhead compared to MIPv4.
- MIPv6 is decoupled from any particular link layer, as it uses IPv6 neighbor discovery instead of Address Resolution Protocol (ARP). This also improves the robustness of the protocol.

- The use of IPv6 encapsulation and the routing header, removes the need in MIPv6 to manage "tunnel soft state".

The MIPv6 supports multiple CoA and may obtain new addresses by stateless or stateful address auto configuration, Dynamic Host Configuration Protocol Version 6 (DHCPv6). These features are of great use in cellular wireless environments where a MN can be reachable through multiple links at the same time. The MN will only register one CoA to the HA and by registering its primary CoA it will keep itself reachable. Upon link change, i.e. the current cell's signal is weaker than another reachable cell's signal, the MN advertises its new primary CoA to the HA with a binding update message, but keeps accepting packets on its old CoA for as long as the old link is reachable or for other determined time (RFC 3775. 2004. draft-ietf-mext-rfc3775bis-03.txt. 2009).

In combination with not having a FA in MIPv6, all above mentioned features enable MIPv6 to conduct smoother handovers between networks, to be a more robust protocol, to have better security for the HA advertisement, and to have less bandwidth overhead than MIPv4.

3.2 VPN over MIP

MIP was originally an extension to the existing IP. This means that it is not in any way cryptographic on its own. Even though there has been a lot of development and extensions done for the MIPv6, it is also just an unprotected way of transporting IP packets to MNs located inside or away from the home network. To add the authentication and encryption demanded to set up a VPN connection some software for authentication and data encryption has to be used. There are three different methods for authentication and data encryption approved by the VPN consortium. Two methods utilize IPsec, IPsec with encryption and IPsec inside L2PT. IPsec with encryption in either tunnel or transport mode is the most used method and supports all requirements for a VPN connection. The security associations can be set up by IKE/IKEv2 or manually. IPsec and IKE/IKEv2 are also well described in many RFC documents, IPsec and L2TP (Layer two Tunneling Protocol) are described in RFC 3193.

3.2.1 MIP and IPsec

IPsec with encryption is one method of setting up a VPN connection over Mobile IP. As Mobile IP keeps track of and forwards packets to the MN, the VPN software must provide application persistence and roaming. There are commercial software packets providing their own solutions based on MIP and IPsec. Since such software is commercial there is no open documentation on how roaming and persistence are implemented, only promises that these features are implemented.

3.2.2 VPN scenario, MIPv4 with two HAs

The Secure Universal Mobile (SUM) architecture and RFC 5265 both address the problems described in RFC 4095 (Problem Statement: Mobile IPv4 Traversal of Virtual Private Network (VPN) Gateways). The solution presented here which is based on the SUM architecture solves the problem more loosely than the defined solution in RFC5265 which strictly follows the outlined solution restrictions and methods in RFC4095. The SUM network setup proposes the same network setup as defined in RFC5265, which suggests the usage of two HA and a De-Militarized Zone (DMZ). This architecture is proposed to support seamless roaming throughout heterogeneous wireless networks both internally and externally. The functionality of the SUM architecture is examined in the following chapter.

Architectural overview

The SUM architecture uses two HA's. One HA is placed in the internal private network and the other is placed in the DMZ. Through this new architecture SUM hopes to;

- Maintain secure reachability to the Intranet by the MN.
- Provide similar security that the user would have inside the private network.
- Provide session continuity for the MN to minimize handoff delays and transient data loss.
- Enable dynamic VPN-tunnel management, so that the MN can communicate on the most cost effective way in terms of security needed and network overhead.

Figure 10 shows a traditional network layout. Figure 10 can be compared to the SUM architecture in Figure 11. Figure 11 shows the SUM network layout with Mobile IP as network layer protocol.

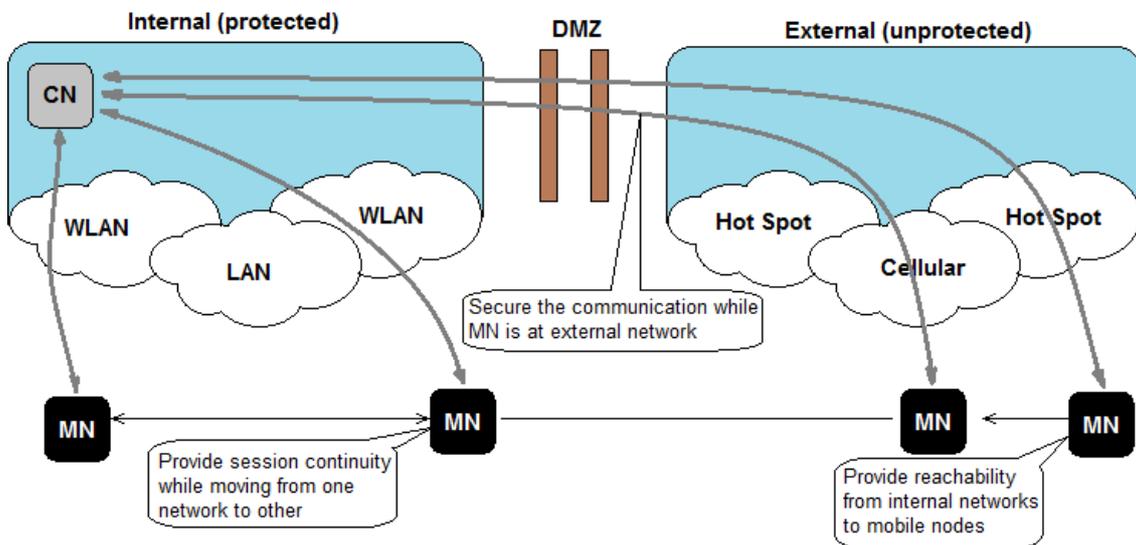


Figure 10. A traditional network layout.

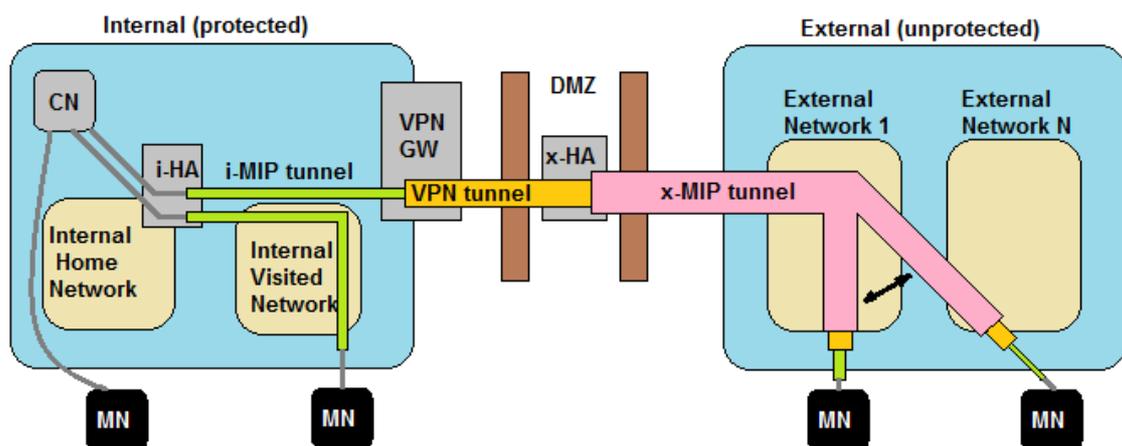


Figure 11. The SUM architecture on the same kind of network with Mobile IP as network layer protocol.

The internal HA (i-HA) is used to support mobility in the internal network. The external HA (x-HA) supports the external mobility and ensures that the VPN session to the MN is kept alive, so that the MN seamlessly can change its topological position in the Internet. Two HA's gives the MN two Home Addresses (i-HoA and x-HoA) and co-located CoA (i-CoA and x-CoA). Each address is assigned by or communicates with the corresponding internal or external HA. The MN will have two instances of MIP running, one internal and one external corresponding to the above mentioned HoA and CoA. The notation for the internal is i-MIP and the external is x-MIP. The VPN address will be noted as TIA (Tunnel Inner Address).

Mobility

The two key features SUM provides are seamless mobility and dynamic VPN-tunnels. The seamless mobility can be explained with the scenarios of the MN moving between the internal and external networks and vice versa, and by the MN moving around between different external networks. Whatever the case is, the roaming is to be done without breaking the VPN connection. For this to succeed four key steps has to be defined:

- The MN registers its new x -CoA to x -HA.
- A VPN tunnel is established inside the DMZ between the VPN gateway and the x -HoA.
- The gateway end of the VPN tunnel is registered as the i -CoA with the i -HA. This allows the i -HA to tunnel packets sent to the i -CoA of the mobile device to be sent through a VPN tunnel to the VPN gateway. The VPN gateway will then tunnel the packets through the VPN tunnel and the x -MIP tunnel so it securely can reach the MN.
- The last step is when the MN moves back into the private network. Then the MIP and VPN tunnels are dismounted and broken down since they are not necessary any more. This might generate some packet losses and delay time, but most applications nowadays have features to cope with this.

Figure 12 shows how the different protocols are interacting in the SUM architecture.

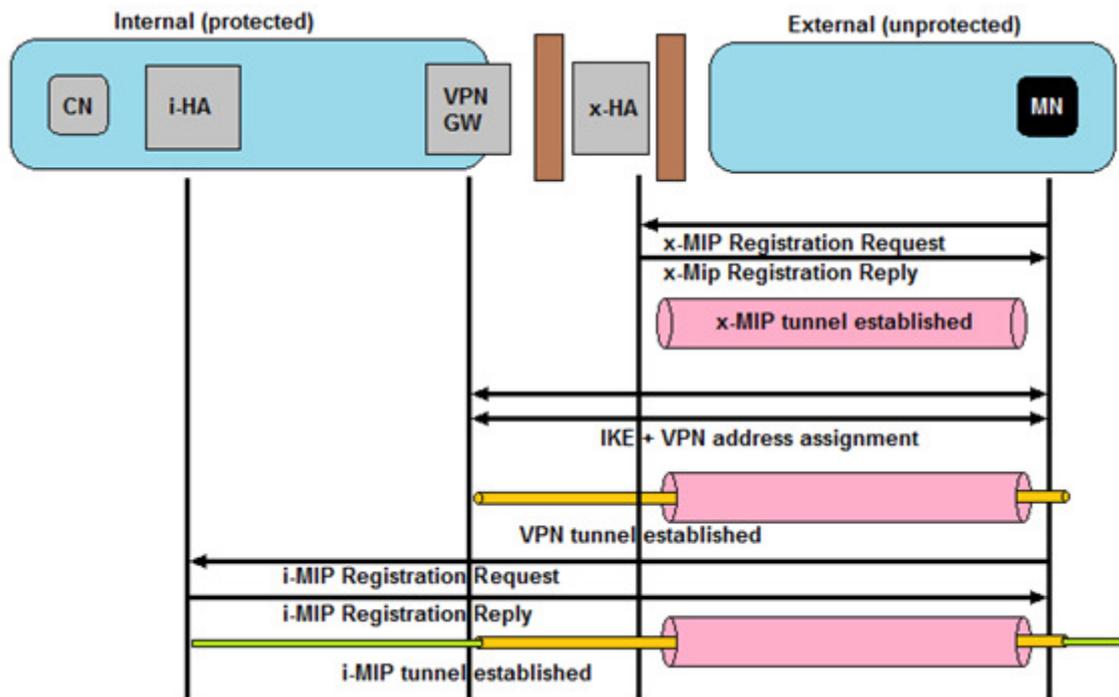


Figure 12. Interaction of protocols for SUM using Mobile IP based architecture.

One of the major concerns with seamless mobility is to minimize handoff delays and data loss in handoffs when the MN is roaming between networks. Therefore SUM utilizes handoff pre-processing and make-before-break strategies to reduce the impact of the delays and data loss. The MN will prepare itself beforehand to the change of networks. The rules for changing networks are not determined by SUM but they could theoretically be determined by the Signal to Noise Ratio (SNR), bandwidth, or cost of the visited and current network. When the MN assumes a change is eminent, preoperational steps are taken;

- Activating its target interface in case of it being shut off when not used,
- Obtaining the new IP-address and other IP-layer configurations,
- Performing any authentication procedure required by the new network, and establishing the network connection needed.

The concept of dynamic VPN tunnels was invented to reduce and remove network overhead and security risks produced by always-on VPN tunnels in mobile networks. The idea is to only set up VPN tunnels when they are needed; when the MN needs to contact/, or when it needs to be contacted by, or when it needs to access the Internet through/ -the internal network. The SUM way of implementing a dynamic VPN tunnel is to utilize the Session Initiation Protocol (SIP) for messaging between the CN inside

the local network and the MN. Two MIP tunnels are set up for SIP messaging, one between the i-HA to the x-HA and another between the x-HA to the MN. The MIP tunnels assure the CN and the MN that the SIP messages reach their destinations. The MIP tunnels are in this stage unprotected. For a CN to initiate a VPN tunnel a SIP INVITE is sent that is either processed by a SIP proxy or is intercepted by the i-HA. Either way the INVITE message is forwarded to the MN over the MIP tunnels. The MN checks if there is an already active VPN tunnel and if not IKE is used to set up a new VPN tunnel. The MN then uses the VPN tunnel to register the gateways end points to the i-HA as described earlier and depicted in Figure 12. Figure 13 illustrates the protocol interaction in a SUM architecture using SIP and Mobile IP.

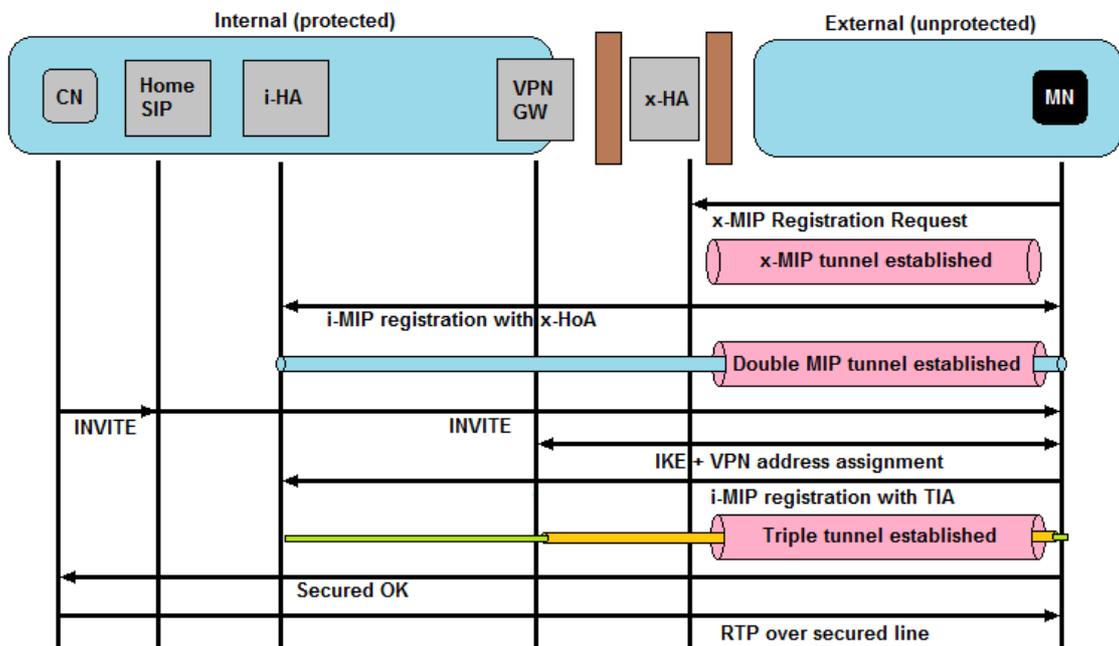


Figure 13. Interaction of protocols in SUM architecture with SIP.

MIPv4, SUM with IKEv2

SUM also proposes an alternative architecture utilizing IKEv2 mobility extensions, MOBIKE. The architecture using MOBIKE would eliminate the costly procedure of re-establishing the VPN tunnel between the x-HA and the MN when the MN changes its IP. This architecture would also reduce the overhead created by the i-MIP, IPsec, and x-MIP triple tunnel described in Figure 14. The key advantage of using MOBIKE is that it can modify the current VPN tunnel SAs and update its endpoint IP address without renegotiation. This feature renders the x-MIP tunnel useless and therefore reduces the

overhead. However, MOBIKE cannot provide by itself seamless mobility when moving between the internal and the external environment. Figure 14 shows the network layout as described using MOBIKE, Mobile IP and SIP.

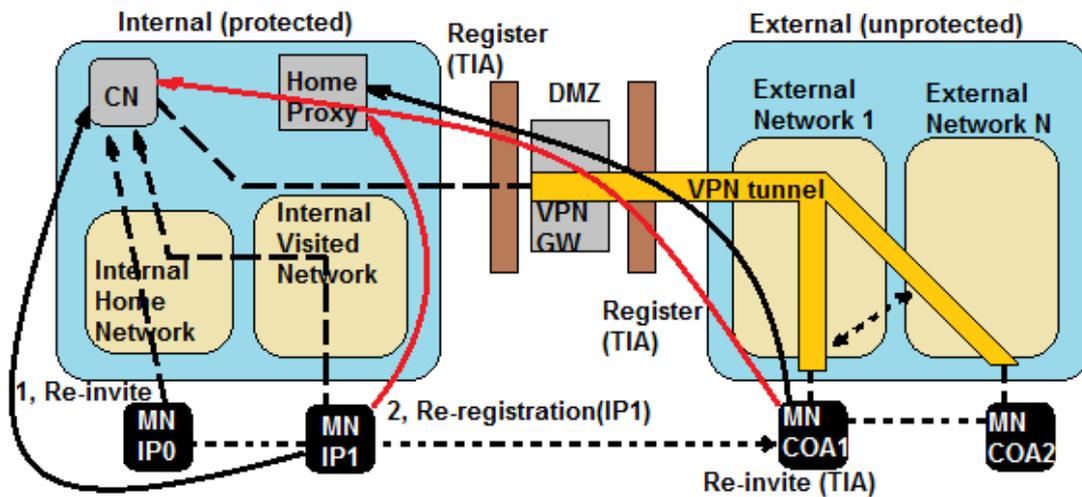


Figure 14. Implementation of the SUM architecture with MIP and SIP.

3.3 HIP

The motivation for HIP and host identities is that a new namespace is needed, since IP addresses are used both as topological locators and as network interface identifiers. The double use of the IP namespace makes mobility very inflexible and unsuitable.

Host identity namespace is a new namespace that will decouple and interoperate with the already existing namespaces IP and DNS. A name in the Host Identity namespace is a Host Identifier (HI), which is a globally unique name that can be assigned to any system with an IP stack, but is not limited to only IP. A host is neither limited to only one HI. A HI is preferably a public key of a ‘public/private key pair’ and should also be initially authenticated with Domain Name System Security Extensions (DNSSEC) or some other third party authenticator (R. Moskowitz & P. Nikander, 2008. RFC4423, 2006).

The Host Identity decouples the topological locator from the interface identifier. In practice this means that the IP addresses are used only to route packets and the network socket is bound to the Host Identifier referring to a single Host Identity. The decoupling of the internetworking and transport layer will also allow the layers to evolve independently. The HI can also be used as a host authenticator since it is a public key,

which can be utilized in authentication by other security protocols such as IPsec. Figure 15 depicts how the new layer is supposed to be introduced to. (R. Moskowitz & P. Nikander, 2008. RFC4423, 2006).

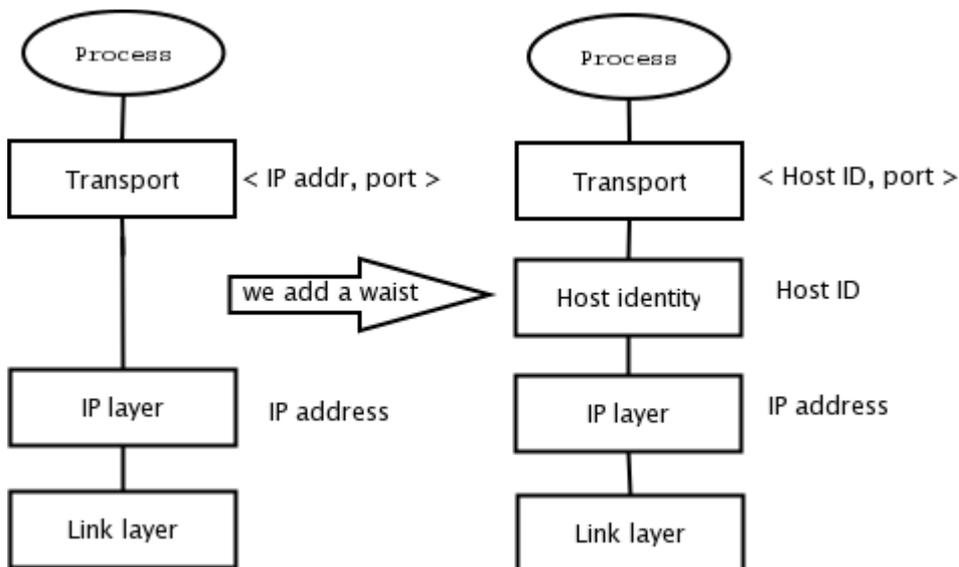


Figure 15. The new Host Identity layer is introduced to the OSI model to decouple the transport layer from the IP layer (InfraHIP Project: Intro, 2009).

A HI is never used directly in the Internet to identify a host, HIs are only used in the HIP Base Exchange. To identify hosts a cryptographic hash is applied to the individual HI to create a 128-bit Host Identity Tag (HIT). The length is fixed because of conveniences for easier protocol coding and for being able to present a consistent format independent of what cryptographic algorithm is being used. A 128-bit cryptographic hash is allocated from the individual HI. Local Scope Identifiers (LSI) is another way of identifying a HI. The LSI is a 32-bit localized representation for a HI. The LSI serves as a bridge for Host Identities into IPv4 based protocols and application programming interfaces (API). LSIs are not globally unique due to their insufficient length (R. Moskowitz & P. Nikander, 2008. RFC4423, 2006).

The cryptographic nature of the HIs is suitable for many authentication systems but a new protocol, HIP is introduced. Older authentications methods such as IKEv2 do not support middle boxes or mobility in the extent HIP can provide. HIP introduces a new cryptographic exchange, host identity Base Exchange. This new exchange enables the peers to set up SAs used by IPsec for setting up an end-to-end secure connection. The data traffic of this connection is secured by taking advantage of ESP packets. Presently, ESP is the best way to secure the payload in a datagram, but in the future better ways to

secure data will surely be developed. HIP also adds multi-homing and mobility features to the enabled HIP devices (R. Moskowitz & P. Nikander. 2008).

The practical set-up of HIP is basically two roundtrip Diffie-Hellman key exchanges, a Base Exchange, and some additional messages. The Base Exchange is done to confirm that two peers actually have own private keys corresponding to own HIs, which are public keys. When a host is authenticated the Base Exchange sets up two SAs to enable the end-to-end security of ESP enveloping. If two hosts need more than two parallel SAs they should use separate HITs for setting up the communication channel. The Base Exchange is illustrated in Figure 16. (R. Moskowitz & P. Nikander. 2008).

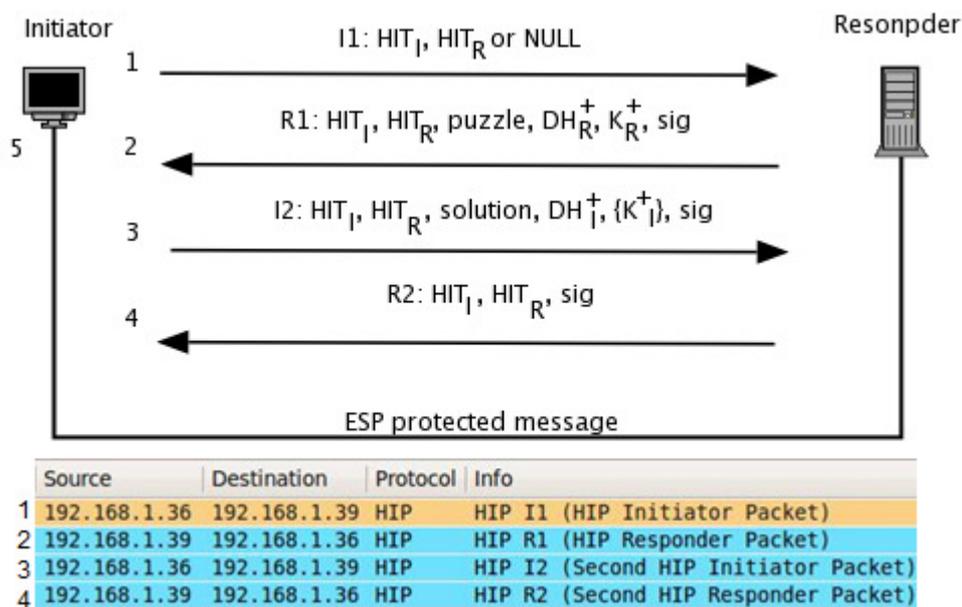


Figure 16. HIP Base Exchange from the initial contact to transmission of the first ESP protected message is sent, as proposed (InfraHIP Project: Intro, 2009).

The flow of the Base Exchange can be described in the following steps (InfraHIP Project: Intro, 2009):

Initiator --> Directory: lookup Responder

Initiator <-- Directory: return Responders's address and HI/HIT

I1 Initiator --> Responder (Hi, Here is my I1, let's talk with HIP)

R1 Responder --> Initiator (Ok, Here is my R1, handle this HIP cookie)

I2 Initiator --> Responder (Computing, here is my counter I2)

R2 Responder --> Initiator (OK. Let's finish HIP with my R2)

Initiator --> Responder (ESP protected data)

Responder --> I (ESP protected data)

A node is mobile when it can change its topological location in the Internet. Changing the location also means that the IP address is changed. A node is multihomed when it has more than one simultaneous connection to the Internet. The node has multiple routes for incoming and outgoing traffic using different interfaces. (HIP Implementation for FreeBSD, 2005). Because the SAs are not bound to IP addresses, a host node is able to receive HIP created ESP SA packets from any IP address. A node can change its topological location and continuously send and receive packets to and from its peers (InfraHIP Project: Intro, 2009).

3.3.1 VPN over HIP

To use HIP as the transport protocol for VPN provides a secure and easy way to a mobile VPN. HIP supports multihoming, which is an essential part for the so called road warrior. A road warrior is a remote user who accesses the corporate intranet from various locations in the Internet. This has traditionally been done with various tunneling methods, such as PPTP or L2TP over IPsec (P. Nikander, 2004). Unfortunately these protocols or combinations of protocols produce a large network overhead and by themselves do not support multi-homing. Therefore they are not suitable for mobile use. The traffic efficiency is critical since mobile devices often connect to the Internet through GPRS or 3G/UMTS networks and extra overhead is costly for the end user. For the session not to be reinitialized every time the network attachment is changed, the multihoming ability is a key feature for mobile VPN implementations.

3.3.2 HIP Proxy

As a result of the HIP Base Exchange, a HIP connection can provide similar security as IPsec without IKE. Therefore placing a HIP proxy between a private network and the Internet will provide the security needed for a VPN connection. The proxy can of course be placed in a public network as well but then the reason to use it is lost. The security can be maintained all the way between HIP enabled host via the proxy to the legacy host if the HIP proxy secures the connections from itself to the legacy hosts in some other way, for example with Secure Shell (SSH) or IPsec.

For a network and a HIP enabled host to work with a HIP proxy the HIP proxy and the HIP enabled host must set up a connection. This is done by configuring the HIP enabled

host to use the HIP proxy as shown in Figure 17. When the HIP enabled host and the HIP proxy sets up a connection, the Base Exchange is done, and the SA for ESP is created, then this HIP enabled host can send ESP protected packets to the HIP proxy. The proxy unpacks the ESP packets to IP packets and sends them to the legacy host. When the legacy host answers to the HIP host with a IP packet then the HIP proxy envelopes it in an ESP packet and sends it to the HIP enabled host. For a host-to-host HIP secured connection also the legacy hosts must change their OSI stacks to support HIP.

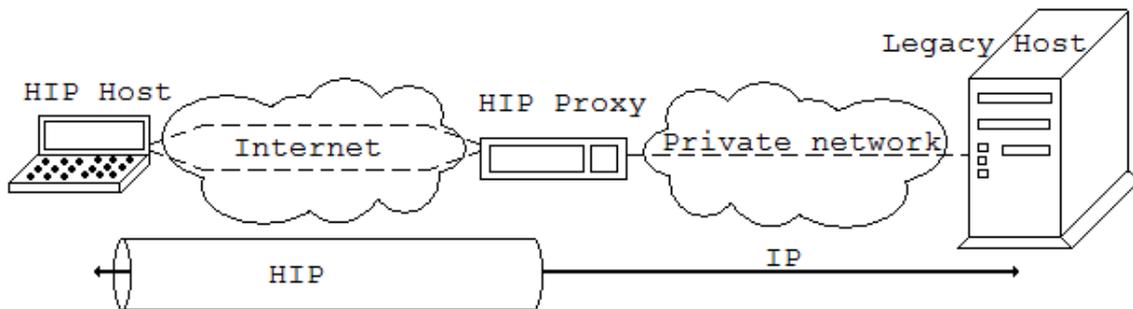


Figure 17. The setup for a HIP proxy between a private network and the Internet.

It is also a cost efficient solution to give all legacy hosts in a private network the ability to access HIP hosts outside the private network through the proxy, since in the private network all hosts are automatically considered to be secure and trusted. All communication in the private network is therefore by itself secure. The proxy can also be used the other way round so that a HIP enabled host from the Internet can access the inside of the private network. This requires the proxy to store a record of trusted HITs and HIs that are allowed to access the intranet (Patrik Salmela and Jan Melén. 2006).

3.3.3 Overlay Convergence Architecture for Legacy Applications

Overlay Convergence Architecture for Legacy Applications (OCALA) is a generic proxy that can be used as a HIP proxy. OCALA introduces a new layer called Overlay Convergence (OC) to the OSI stack, layered under the transport layer. The OC layer bridges legacy applications and overlays (HIP). The OC provides an IP like interface to the legacy applications and then tunnels the traffic over the overlays. The OC is divided into a OC-I sub-layer and a OC-D sub-layer. The OC-I providing the IP like interface to the legacy applications and the OC-D that is communicating with the overlay. The overlay consists of modules that implement different protocols for tunneling purposes,

protocols such as HIP, i3, RON and other. The overlay provides a way to implement any protocol through the OC-D sub-layer. Figure 18 illustrates the two sub layers and their relation to the Transport layer and the different plug-ins (OCALA - Overlay Convergence Architecture for Legacy Applications. 2006).

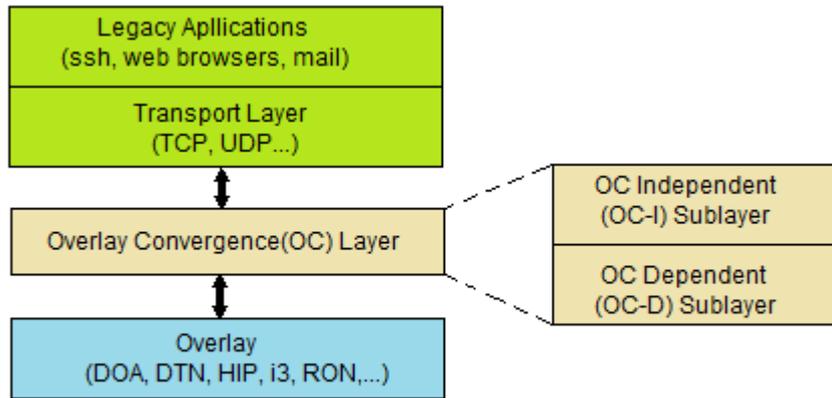


Figure 18. The OCALA architecture, OC layer is placed under the transport layer.

OCALA Proxy

OCALA can be used as a proxy for the different overlay protocols, giving legacy host the ability to benefit from the overlay protocols features. The proxy must be implemented in a secure way to benefit from the security features HIP can provide.

A common scenario is that a legacy host running a legacy application inside a private network wants to communicate with a mobile host over HIP. The OCALA proxy provides an interface between the private network communicating with IP and the mobile host communicating with HIP as depicted in Figure 19. The OCALA proxy tunnels the IP packets through the overlay HIP module. The mobile HIP host receives the packets and supplies the legacy host through the IP like interface OC-I provides. The result is that the legacy software on the legacy host inside the private network can communicate over HIP with a mobile host's legacy software unaware of the OCALA proxy. This scenario is also working the other way around, with the mobile host communicating with hosts inside the private network (OCALA - Overlay Convergence Architecture for Legacy Applications. 2006).

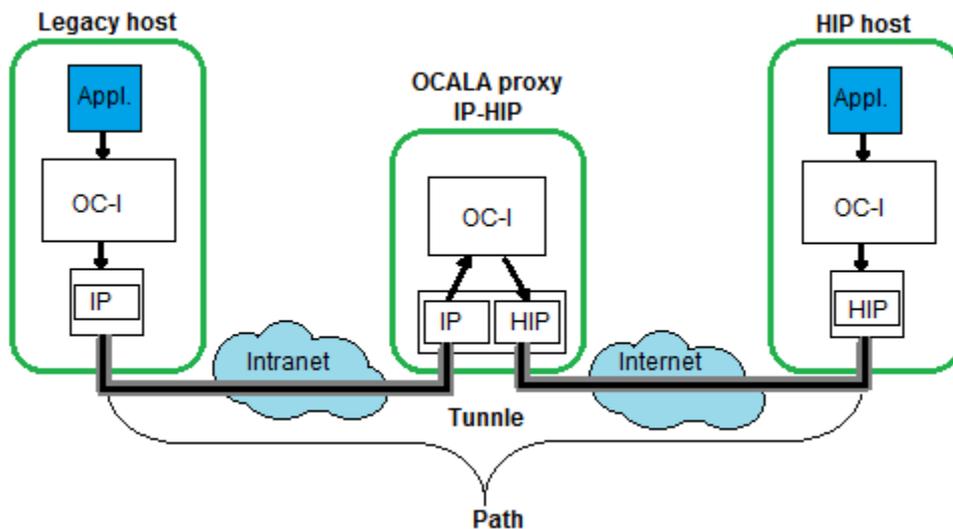


Figure 19. A legacy host running a legacy application communicating with a mobile HIP host's legacy application through a OCALA IP-HIP proxy.

3.3.4 HIP VPN Gateway

To enable a mobile host, a road warrior, to access a private network using HIP a HIP gateway implementation is preferable. A HIP gateway set up the correctly gives the road warrior the best HIP solution for accessing his private/corporate network. Legacy hosts inside a private network that wishes to communicate with a mobile HIP host, must know where in the Internet the MN is located. This can be implemented with a so called Rendezvous server (RSV). When the legacy host knows where in the Internet the MN is located it can either use a HIP proxy in its own private network or HIP installed locally for full HIP support.

3.4 Theoretical conclusion

The research on the theoretical background has provided a deeper knowledge in the examined protocols and technologies. The theoretical part is divided into two main chapters VPN Implementations and Mobility Protocols. All protocols and technologies examined in both chapters are suitable for mobile VPN implementations.

The VPN part of the chapter examines the IPsec, MOBIKE and SSL/TLS protocols and their usage in VPN scenarios. They are all possible candidates for successful usage in mobile VPNs. IPsec with or without MOBIKE is suitable because of its already broad

usage in VPN implementations and its durability. SSL/TLS VPNs are suitable because of lightweight, small network overhead, ease-of-use, and also many features providing more personal and precise rules for users.

The Mobility Protocol chapter examines MIPv4 and MIPv6 with more focus on version 6 as well as HIP. Both protocols are developed with the mobility in mind and they are both theoretically good candidates for providing features desired in mobile environments. Multihoming, seamless roaming between networks, and continuous accessibility are features which these protocols offer the user.

Theoretically, implementing these protocols and technologies correctly should provide mobile VPN with a high level of usability, security, performance, and mobility.

4 EVALUATION OF MOBILE VPN SOLUTIONS

4.1 Birdstep SafeMove

4.1.1 Overview

Birdstep is an Scandinavian company founded in 1996 and with its headquarters in Oslo, Norway. Birdstep software SafeMove 5.0 is a Mobile VPN solution utilizing Mobile IP, IKE, and IPsec. SafeMove offers the end user seamless mobility, session persistence, and genuine .zero-click connectivity. The SafeMove software can be installed on computers, smart phones and PDAs.

4.1.2 Mobility

BirdStep is using Mobile IP as the mobility protocol. BirdStep has implemented eleven RFC documents concerning Mobile IP. Among other the RFC 3519 dealing with NAT traversal is implemented to deal with NAT traversal issues. BirdStep also supports thirteen different RFC standards dealing with VPN security, most of these deals with IKE versions one and two and IPsec. BirdStep has built a solution around IPsec and IKE. The used RFC standards are implemented to meet the mobility and security demands (Birdstep Technology Oy, 2009).

For the SafeMove solution to work a client software has to be used by the MN. The client software is found for Windows, Linux and Apples OSx. This gives SafeMove an advantage, since most other Mobile VPN solutions are not as versatile. SafeMove architecture depends on a legacy server provided by BirdStep (Birdstep Technology Oy, 2009).

4.1.3 Conclusion

BirdStep offers a complete Mobile IP solution with a lot of supported standards, which ensure that it will work well with legacy networks. However, BirdStep seems not to be willing to share any of the information regarding the network topology or other changes that have to be done to a network were the SafeMove solution is to be implemented.

Furthermore, BirdStep holds many patents regarding solutions for dealing with Mobile VPN (Birdstep Technology Oy, 2009).

4.2 Cisco Mobile VPN

4.2.1 Overview

Cisco Systems released 2006 a whitepaper about a mobile VPN based on Mobile IP and Cisco Mobile Client software (Cisco Mobile VPN, 2006). Cisco utilizes the features that Mobile IP provides in own hardware and software. The end user can enjoy seamless mobility and session persistence, which in combination with IPsec result in a VPN connection. The white paper referees to RFC 2794 Mobile IP Network Access Identifier Extension for IPv4 from the year 2000. Cisco Mobile VPN uses MIP as specified in another white paper by Cisco Systems (Cisco Mobile IP, 2001). This white paper refers to one of the earliest RFC's concerning mobile IP (RFC 2002, 1996). However, it is assumable that Cisco has updated own standards along the way to support newer versions of the MIP protocol.

To add seamless mobility, Cisco Mobile VPN interacts with the client's network hardware to retrieve information about signal strength. The client software uses this information to determine if the signal strength is under a certain threshold. If the signal is low, the client software sets up an alternative connection and prepares itself to switch if the signal strength falls under a second threshold. The threshold may be set to other parameters than signal strength e.g. to cost, bandwidth, or it can be manually configured.

4.2.2 Network scenario

Cisco Mobile VPN network setup is not too different from the standard Mobile IP network setup. Cisco uses a HA to route the packets to the MN and to keep track of the MN's movement. However, Cisco Mobile VPN doesn't use an separate FA as supposed for Mobile IPv4 networks. Instead Cisco has developed client software that has taken the FA's place. This innovation gives the "road warrior" the desired uncomplicated

setup. Furthermore, an FA can also be introduced to this solution to increase scalability for the “road warriors” network.

4.2.3 Cisco Mobile VPN specific configuration example

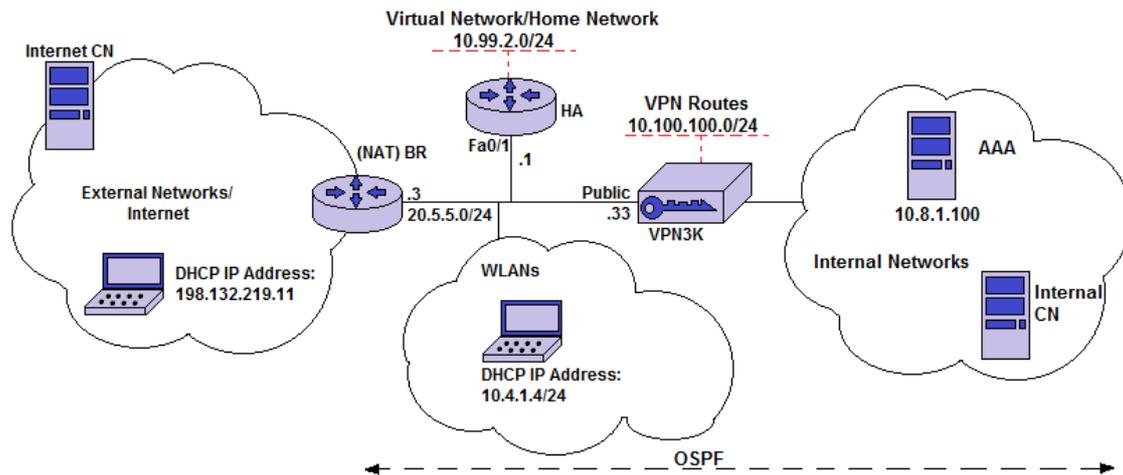


Figure 20. Cisco Mobile IP VPN configuration Topology Example.

There are some specific configurations that need to be considered for the Cisco Mobile IP VPN. Figure 20 depicts a network with Cisco Mobile VPN implemented:

- HA addresses must be publicly accessible. This feature gives the MN the ability to move from the internal network to an external network.
- In AAA authentication the IPsec gateway must permit UDP on port 1645. This is critical if the HA uses SA’s stored on an AAA server located inside the private network.
- If NAT is used by the VPN gateway to assign the MN IP addresses and the MN needs to reach the Internet, then traffic from the MN would need to be NATed to the public IP address.
- Mobile IP Reverse tunnel features must be enabled. This is done in order to bypass a potential Reverse Path Forwarding check. This feature is used in most ISP networks.
- The IPsec tunnel must run on top of Mobile IP tunnels. No changes to the IPsec tunnel endpoint are done.

4.2.4 Conclusion

The Cisco Mobile VPN solution seems like a solid Mobile VPN solution. The usability seems to be high since the only interaction demanded by the “road warrior” is to start up the client software and authenticate him/herself. Since this is a supplier specific solution, the network side seems to be solid and tested enough to meet the demands of what a mobile VPN environment is demanding. However, since this is a supplier specific solution it may require big investments for a organization to be able to implement it.

4.3 IETF standard for Mobile IPsec VPN based on Mobile IPv4 and MOBIKE

The IETF document RFC 5266 outlines a mobile IPsec VPN solution based on Mobile IPv4 and MOBIKE. There is only one HA, an i-HA in the internal network, and only co-located care-of addresses (co-CoA) can be used in the external network. The resulting network topology is depicted in Figure. 21. A MN has thus only one instance of Mobile IP running, the internal i-MIP.

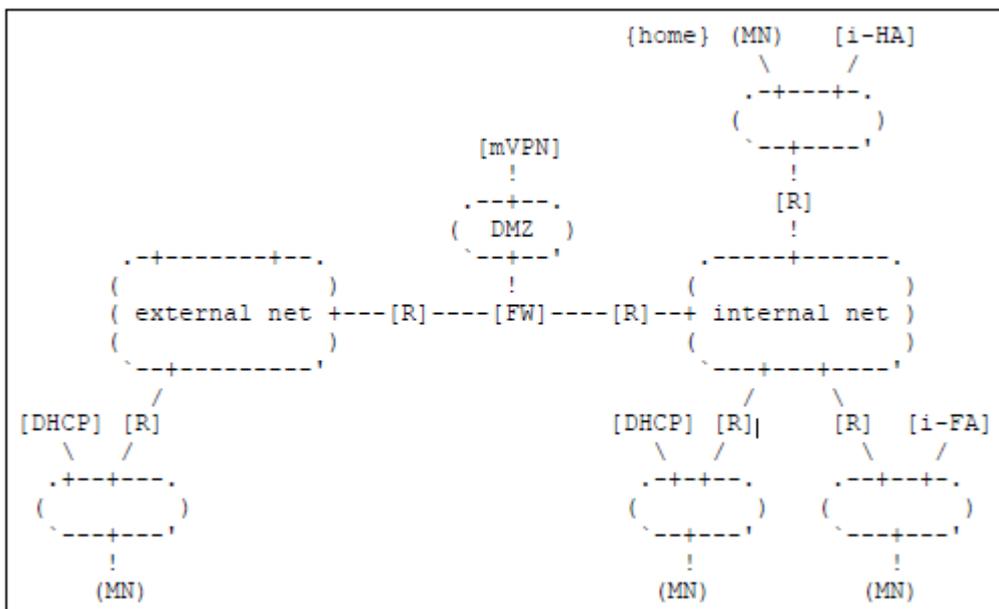


Figure 21. Mobile IPsec VPN based on Mobile IPv4 and MOBIKE (R=router).

When a MN is located in the internal network, the current i-CoA or FA-CoA is registered in the i-HA and the MN has only i-MIP running. The MN may have valid

IKEv2 SAs with the VPN gateway. An IPsec SA can be created when required. When the MN moves to a new network attachment point, then the following four steps are taken:

- The MN obtains a co-located x-CoA or i-CoA from the local network.
- At the same time,
 - o a MOBIKE exchange to update SAs of an existing IPsec VPN connection with the new MN IP address is initiated. If no valid VPN connection exists, then an IPsec VPN connection between the new MN IP address and VPN gateway is established
 - o the MN sends a Mobile IPv4 Registration Request to the i-HA without VPN encapsulation.
- If the MN receives a Registration Reply, then the MN is located in the internal network and the MN can communicate without VPN tunneling.
- If the MN receives no response to the Registration request and if a valid VPN tunnel exists, then the MN sends a Mobile IPv4 Registration Request to the i-HA through the VPN tunnel. After receiving a Registration reply from the i-HA, the MN is ready for data communication through the VPN tunnel.

After these four steps, IP packets sent to Home Address of the MN are tunneled through the VPN gateway to the MN. When the MN after this roams in the external network, then all four steps are needed each time the point of network attachment is changed. If the MN moves from the external network to the internal network, then only the three first steps are needed.

Successful roaming behind, to or from behind a NAT gateway is achieved in the same way as for an IPsec VPN based on only Mobile IPv4. Depending on the MIP access mode used and possible placement of a NAT device, different NAT related problems are eminent. Different solutions to eliminate problems associated with NAT devices must be used. The problems are the same as with Mobile IPv4 without MOBIKE.

4.4 Mobile IPsec VPN based on Mobile IPv6

The mobile IPsec VPN solution based on Mobile IPv4 specified in the IETF document RFC 5265 also works for Mobile IPv6. The mobile IPsec VPN solution based on

Mobile IPv4 and MOBIKE specified in the IETF document RFC 5266 also works for Mobil IPv6 implementations with MOBIKE support. An open source IPsec implementation with MOBIKE support for both IPv4 and IPv6 is StrongSwan (www.strongswan.org). A difference in comparison with mobile IPsec VPN solution based on Mobile IPv4 is that a MN in Mobile IPv6 can have only co-located care-of addresses. The performance of IPsec VPN solutions based on Mobile IPv6 may however suffer from two instances of IPsec tunneling, since signal messages between a MN and it's HA are IPsec protected.

5 PRACTICAL STUDIES

5.1 Test environment and tools

5.1.1 VMware Player

VMware Player is free software for running virtual machines on a computer. VMware Player is developed by VMware Inc. which is the leading company for developing virtualization software. VMware offers a large software suite that covers a broad spectrum of tools used in virtualization. VMware Player is one of these tools.

A virtual machine is isolated software simulating a real machine; such a machine can run its own operating system and applications as a real computer. A virtual machine has its own virtual hardware, everything from a Central Processing Unit (CPU), Random Access Memory (RAM) to network adapters and hard drives. A virtual machine behaves exactly as a real computer, applications and operating systems are unaware of running on software implemented hardware. Not even the virtual machine itself knows that it is virtual. These features and the fact that everything is running on virtual hardware are all things that are advantages over physical hardware. Such advantages are compatibility, isolation, encapsulation and hardware independence (Virtual Machines, Virtual Server, Virtual Infrastructure – VMware: Virtualization Basics, 2009).

Compatibility, means that the virtualization software works with all x86 operating systems and all hardware found in a real computer. Any operating system or software that runs on a real computer runs also on a virtual component.

Isolation, means that hardware of a computer running multiple virtual machines is shared but also isolated from the hardware use of other virtual machines. This means that if one of the virtual servers crashes, then this crash will not affect any other virtual machines. The crashed virtual server can be restarted and served while the other virtual machines are running normally.

Encapsulation, means that a virtual machine is a package of software implemented hardware, an operating system, and applications. The package is portable and can be copied and moved to any other machine running VMware Player.

Hardware independence means that the virtual hardware is completely independent of the physical computer. The virtual hardware can be configured and tuned in any way and is not inflicting danger to the physical hardware. Hardware independence also enables portability of a virtual machine as long as the x86 architecture remains.

VMware Player is used in this B.Sc. thesis project to test applications in a network, in which it is easy to switch between operating systems and between different configurations of programs and operating systems.

5.1.2 Ubuntu

Ubuntu is a community developed Linux distribution. Because of easy installation and a package manager, Ubuntu is one of the most used Linux distributions for personal computers. Ubuntu is a free of charge operating system providing versions suitable for different server applications and for regular desktop workstations. Ubuntu is available as long time service (LTS) versions or as regular versions. In both cases Ubuntu will always stay free of charge either you are installing a new version or upgrading an old. Ubuntu supports the best and newest software that the open source community provides with its easy-to-use package handler. Since the first release of Ubuntu in October 2004 there have been many improvements. Development rate is still high. The goal is to achieve a new version release every 6 months (What is Ubuntu? | Ubuntu, 2009. The Ubuntu Story, 2009).

Ubuntu is the chosen Linux distribution for this B.Sc. thesis project because it is easy to install, to manage, and to keep up to date. Ubuntu is also one of the most supported Linux distributions for ready built packages. These features make installation of components and programs easy. Moreover, there are never any compilation problems, since the packages are binaries.

5.1.3 OpenVPN

OpenVPN is an open source cross-platform SSL VPN solution with all features that any other competitive VPN solution provides. OpenVPN supports a wide range of configurations modes including remote access, site-to-site VPN, Wi-Fi security, enterprise-scale remote access with load balancing, failover, and fine-grained access

controls. OpenVPN is lightweight compared to other VPN solutions and is based on the well known and widely used SSL/TLS protocol providing security on the transport level of the OSI stack. OpenVPN provides a variety of authentication methods based on certificates, smartcards, and devices providing strong authentication i.e. two factors. Due to the variety of authentication methods, user specific and/or group specific access control policies can be implemented through firewall rules applied to the virtual interface created by OpenVPN (About OpenVPN,2009).

OpenVPN is written as a user-space daemon, not as a kernel module as IPsec is implemented. The daemon running in the user space gives to OpenVPN an easy installation and configuration. As a premise states “complexity is the enemy of security”, the simplicity is a good thing. Running in user-space also makes OpenVPN easy to port and implement in different environments. OpenVPN uses an industrial strength security model designed to protect against both active and passive attacks. The model is based on SSL/TLS for session authentication and on the IPsec ESP protocol for secure tunnel transport over User Datagram Protocol (UDP) and TCP. OpenVPN does not support IPsec even if it uses the ESP protocol. There is also no support for IKE, PPTP or L2TP. OpenVPN is compatible with RSA certificates and X509 public key infrastructure (PKI), NAT, DHCP, and TUN/TAP virtual devices. OpenVPN is not a web application proxy, it is a full range VPN solution requiring installation on both server and client (About OpenVPN,2009).

OpenVPN is used in this B.Sc. thesis project as VPN software because it is easy to install and use. OpenVPN is also available as binaries for Ubuntu.

5.1.4 Wireshark

Wireshark is a protocol analyzer and it is the leading of its kind. Wireshark is open source, and it has been under development since 1998 by security and network specialists from all over the world. A wide range of protocols are supported. Because Wireshark is open source it is possible to make own plug-ins for unsupported protocols. It has a good graphical user interface and runs on most operating systems. It also provides good tools for filtering, for saving output files, and for reading files in different formats (Whireshark: About, 2009).

HIP is a protocol that is not supported in the original Wireshark protocol suite. Wireshark must therefore be patched. To patch Wireshark is not a difficult process except for finding the right protocol plug-in version for the right Wireshark version (See Appendix B for a guide on how to patch Wireshark for HIP).

Wireshark is used in this B.Sc. thesis project for analysis and verification of network activity corresponding to the anticipated behavior of different protocols.

5.1.5 InfraHIP project

InfraHIP

InfraHIP is a project in HIIT. Infra stands for infrastructure addressing issues such as DNS, NAT and firewall support to make HIP a widespread protocol. HIP stands for Host Identity Protocol and the basic protocol is almost finished. InfraHIP studies application related aspects of HIP such as rendezvous service, operating system security, multihoming, process migration and issues related to enterprise level solutions. The project collaborates with Distributed System Group in RWTH Aachen (InfraHIP project: About, 2009).

HIPL

HIP for Linux or HIPL is the software developed by the InfraHIP project to provide an application base for the InfraHIP project. HIPL is experimental but well working software and relatively easy to setup and use. HIPL is easiest described with a couple of use case scenarios (InfraHIP Project: HIPL):

- Security for different types of Internet connectivity. A TLS type of security is provided with support for both Internet Control Message Protocol (ICMP) and UDP protocols. Legacy applications are not required to be changed to take advantage of the authenticity, confidentiality and integrity features provided by HIPL.
- Public key based access control. A firewall module supporting and a graphical user interface for end users are provided to enable and make public key access control easier.

- Universal and persistent Internet connectivity. This feature makes it easy for end users to provide web services from behind a NAT without changing the NAT configurations. This feature also enables hosts to keep sessions alive while moving across different networks and points of attachment to the Internet.

HIPL is used in this project because it is the most up-to-date implementation of HIP according to HIP related RFC documentation. It is also released as a binary for Ubuntu making it easy to install and use in an Ubuntu environment. HIPL and InfraHIP also provide good documentation, testing programs and scenarios, and possible implementation examples.

5.2 Mobile VPN with HIP – practical testing

5.2.1 InfraHIP and openVPN

This test is done with HIP as the mobility protocol, and HIPL as the software providing HIP and OpenVPN as the VPN software. The test purpose is to determine if this combination of protocol and software will work, and how well it works considering mobility. The experiment is based on the fact that it is mentioned in the InfraHIP manual as a possible implementation of HIPL (OpenVPN Compatibility, 2009). Mobility will be tested by changing the Internet connection network interface. The assumption is that the session will not be dropped and no renegotiation of the HIP session will be necessary, as long there is at least one active network interface on the MN. The VPN connection is assumed to stay open all the time since both network interfaces will be active and both interfaces will be registered for usage. The same experiment will also be conducted with a delay between the switching of the network to determine how well it will work under a time-out scenario. The assumption is that the HIP session will be renegotiated, but that the VPN session is saved.

The test is done with two computers in the same subnet, since the purpose is to test the concept and to see if it works. Thereby eliminating as many peripheral variables as possible is desirable. The two computers will be identified as Computer (A) and Computer (B), and their IP addresses are, 192.168.1.44, 192.168.1.42 and 192.168.1.43. Computer (A) has the HIT 2001:0019:d7bd:d59c:fda4:eb6e:79fa:cb5b and Computer (B) 2001:001a:592b:6d45:0946:83e5:bfe8:6057. The test starts with

testing the concept by not including mobility. Depending on the success of the concept test the mobility test is executed. The results will be recorded. The OpenVPN configurations can be found in Appendix C.

Some additional configuration is needed to get InfraHIP to use custom LSI's, see configuration guide in Appendix D. After the additional configuration and startup of hipd and hipfw processes OpenVPN can be started. The tunnel interface will be using the addresses 10.8.0.2 for Computer(A) and 10.8.0.1 from Computer(B).

InfraHIP and OpenVPN proof of concept

The proof of concept test was successful, the tunnel was set up using the configured LSI (1.0.0.50) for Computer(B). As shown in Figure 22 the tunnel is successfully set up using LSI of Computer(B). Figure 23 illustrates the packet by packet setup flow during the initialization of the tunnel.

```

ubuntu@ubuntu904:~/Desktop/openVPNs$ sudo openvpn client openVPN.conf
Thu Jan  7 17:19:07 2010 OpenVPN 2.1_rc11 i486-pc-linux-gnu [SSL] [LZ2] [EPOLL]
[PKCS11] built on Mar  9 2009
Thu Jan  7 17:19:07 2010 /usr/sbin/openvpn-vulnkey -q static.key
/usr/sbin/openvpn-vulnkey:22: DeprecationWarning: the md5 module is deprecated;
use hashlib instead
import md5
Thu Jan  7 17:19:07 2010 TUN/TAP device tun0 opened
Thu Jan  7 17:19:07 2010 /sbin/ifconfig tun0 10.8.0.2 pointopoint 10.8.0.1 mtu 1
500
Thu Jan  7 17:19:07 2010 UDPv4 link local (bound): [undef]:1194
Thu Jan  7 17:19:07 2010 UDPv4 link remote: 1.0.0.50:1194
Thu Jan  7 17:19:11 2010 Peer Connection Initiated with 1.0.0.50:1194
Thu Jan  7 17:19:11 2010 Initialization Sequence Completed

```

Figure 22. Computer(A), OpenVPN tunnel successfully initialized. Computer(B) LSI: 1.0.0.50.

No. .	Time	Source	Destination	Protc
356	332.422528	193.167.187.134	192.168.1.43	TCP
357	332.425472	193.167.187.134	192.168.1.43	TCP
358	332.425491	192.168.1.43	193.167.187.134	TCP
359	332.425928	193.167.187.134	192.168.1.43	TCP
360	332.457747	192.168.1.43	193.167.187.134	TCP
361	332.465250	192.168.1.43	193.167.187.134	TCP
362	335.287910	192.168.1.43	217.30.182.230	DNS
365	335.331743	217.30.182.230	192.168.1.43	DNS
366	335.335754	192.168.1.43	217.30.180.230	DNS
367	338.445262	192.168.1.44	192.168.1.43	UDP
368	338.445262	2001:1a:592b:6d45:946:83e5:bfe8:6057	2001:19:d7bd:d59c:fda4:eb6e:79fa:cb5b	UDP
369	338.445759	1.0.0.51	1.0.0.1	UDP

Figure 23. Computer(B), Wireshark log of the OpenVPN tunnel initialization. Computer(A) LSI: 10.0.0.51.

InfraHIP and OpenVPN – mobility

The mobility test failed. Switching between network interfaces caused the VPN tunnel to stop working. However, the VPN tunnel did not crash because it could resume without renegotiation. This proves that InfraHIP makes the VPN tunnel mobile.

Furthermore, since the mobility is proven to work without the hipfw process running, the assumption is that the hipfw is the process causing the mobility problem. However, the hipfw process needs to be running to add the LSI support needed by OpenVPN.

5.2.2 InfraHIP and SSH

Proof of concept and mobility of InfraHIP and SSH are tested. Computer (A) will mount a directory on Computer (B) over SSH inside a HIP tunnel. Computer (A) is identified by IP's 192.168.1.44, 192.168.1.42 and HIT 2001:001a:592b:6d45:0946:83e5:bfe8:6057. Computer(B) is identified by IP 192.168.1.43 and HIT 2001:0019:d7bd:d59c:fda4:eb6e:79fa:cb5b. Computer(B) HIT is added to the hosts file of Computer(A) for convenience. To easily see if there is any interruption in the transfer, a video is opened from the remotely mounted drive. The video has little significance in the proof of concept test. However, in the mobility testing it reveals if there is a problem with the handover between network interfaces on Computer(A) and if the update is done smoothly between the connected computers.

InfraHIP and SSH proof of concept

The proof of concept test is just testing that InfraHIP and SSH work together as expected. Computer(A) connects to Computer(B) by the HIT of Computer(B). The software used to mount a drive over the network is built in for Gnome 2.5 and later versions. Both test computers are running Ubuntu 9.04 which has Gnome preinstalled. Figures 24 and 25 prove that the proof of concept test was successful. Figure 24 shows how the video is played from the remotely mounted hard drive unit. Figure 25 shows the encrypted data.

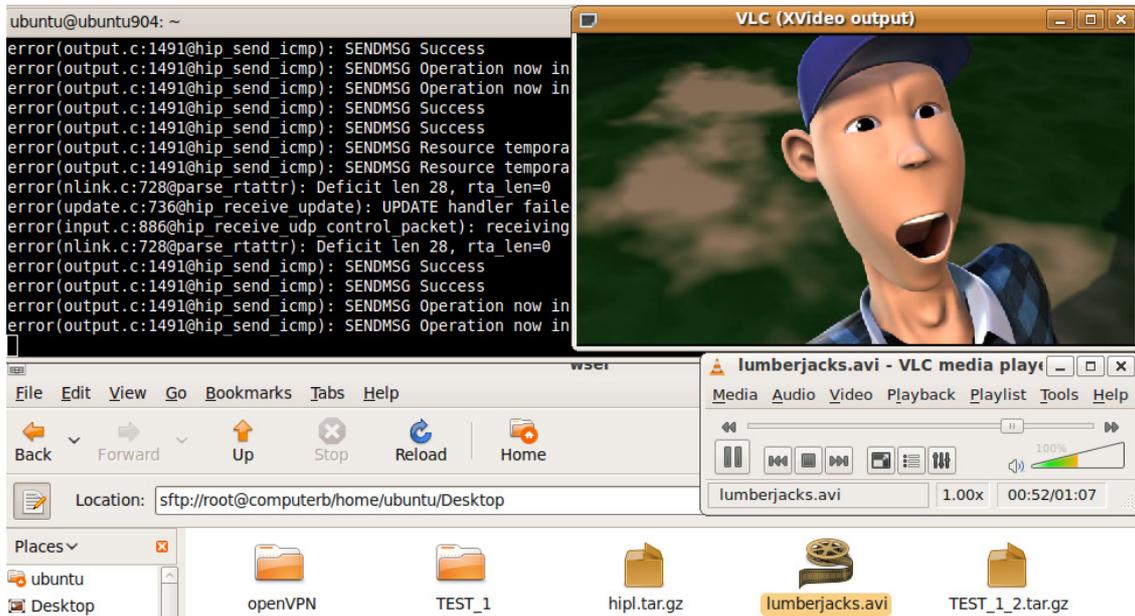


Figure 24. The successful proof of concept test. In the picture; output from InfraHIP process, output from remote video, explorer showing the remote directory where the video is located.

No. .	Time	Source	Destination	Protocol	Info
270	0.856381	2001:1a:592b:6d45:946:83e5:bfe8:6057	2001:19:d7bd:d59c:fda4:eb6e:79fa:cb5b	SSH	Encrypted request packet len=64
336	0.877281	2001:1a:592b:6d45:946:83e5:bfe8:6057	2001:19:d7bd:d59c:fda4:eb6e:79fa:cb5b	SSH	Encrypted request packet len=64
384	0.887744	2001:1a:592b:6d45:946:83e5:bfe8:6057	2001:19:d7bd:d59c:fda4:eb6e:79fa:cb5b	SSH	Encrypted request packet len=48
415	0.976914	2001:1a:592b:6d45:946:83e5:bfe8:6057	2001:19:d7bd:d59c:fda4:eb6e:79fa:cb5b	SSH	Encrypted request packet len=144
420	1.226141	2001:1a:592b:6d45:946:83e5:bfe8:6057	2001:19:d7bd:d59c:fda4:eb6e:79fa:cb5b	SSH	Encrypted request packet len=64
494	1.253494	2001:1a:592b:6d45:946:83e5:bfe8:6057	2001:19:d7bd:d59c:fda4:eb6e:79fa:cb5b	SSH	Encrypted request packet len=48

Figure 25. Successful proof of concept, SSH packets addressed with the HIT of Computer(A).

The proof of concept test was successful, Computer(A) could mount a directory from Computer(B) as a local driver using HIT of Computer(B).

InfraHIP and SSH mobility – mobility

Because of the successful proof of concept test, the same setup is tested for mobility. The mobility test is done in the same setup and environment. The only different is that now Computer(A) will be using both network interfaces. The network interface state is randomly changed from up to down and vice versa. This is done to simulate a road warrior changing between networks. The expected outcome is that the SSH tunnel always stays intact, while the HIP tunnel is changing between the interfaces of Computer(A). Also the scenario of both network interfaces being down will be tested.

No. .	Time	Source	Destination	Protocol	Info
32080	04.244005	192.168.1.43	192.168.1.44	UDP	Source port: 10500 Destination port: 10500
32087	65.802611	192.168.1.42	192.168.1.43	UDP	Source port: 10500 Destination port: 10500
32088	65.805508	192.168.1.42	193.167.187.149	UDP	Source port: 10500 Destination port: 10500
32089	65.905610	192.168.1.43	192.168.1.42	UDP	Source port: 10500 Destination port: 10500
32090	65.905746	192.168.1.43	192.168.1.44	UDP	Source port: 10500 Destination port: 10500
32091	65.978272	193.167.187.149	192.168.1.42	UDP	Source port: 10500 Destination port: 10500
32092	66.001820	127.0.0.1	127.0.0.1	DNS	Standard query SOA hit-to-ip.infracorp.net
32093	66.002771	127.0.0.1	127.0.0.1	ICMP	Destination unreachable (Port unreachable)
32094	66.014773	192.168.1.42	192.168.1.43	UDP	Source port: 10500 Destination port: 10500
32095	66.119581	192.168.1.42	193.167.187.149	UDP	Source port: 10500 Destination port: 10500
32096	66.120371	192.168.1.42	192.168.1.43	UDP	Source port: 10500 Destination port: 10500
32097	66.120962	192.168.1.43	192.168.1.44	UDP	Source port: 10500 Destination port: 10500
32098	66.121221	192.168.1.42	193.167.187.149	UDP	Source port: 10500 Destination port: 10500

Figure 26. Wireshark dump showing successful update of address of Computer(A) from 192.168.1.44 to 192.168.1.42.

Figure 26 shows a successful update of IP address. The data of the video stream is inside the UDP packets. The video was playing all the time except for the time when the update was done. When both network interfaces was set to down the video stopped, since there was no Internet connection anymore. However, when one of the two network interfaces was set active it resumed.

Overall InfraHIP worked quite well, except for some random malfunctions that had to be resolved with restart of the HIP daemon and other software.

5.2.3 OpenHIP and OpenVPN

This test will be investigating OpenHIP as the mobility protocol and OpenVPN as the VPN application. OpenVPN is as described earlier a VPN using SSL/TLS to achieve the demanded security. The combination of OpenHIP and OpenVPN is at this point the best combination on HIP implementation and VPN to achieve mobile VPN and reach a broad user base, because they are both compatible with both Linux and Windows systems and OpenVPN is fairly lightweight as a VPN-client.

The test is a two part test. The first part examines the concept of OpenHIP and OpenVPN working together. The second part tests the mobility. The test is executed on two virtual computers running a Linux system, Ubuntu 9.04. Computer A will pose as the client and it has two network interfaces. Computer B acts as the server and has the same IP throughout the whole test, while Computer A randomly changes between its interfaces in the mobility test. Both machines are running OpenHIP version 0.7 and OpenVPN version 2.11_rc11. The VPN will be tested only by a point-to-point tunnel between the client and the server. This test is enough to prove the concept, since more advanced setups all have this type of connection.

Server and Client configuration files can be found in the Appendix C. However the client uses the locally configured LSI as the remote address. Computer(A) is configured to have the IP 10.8.0.2 and Computer(B) 10.8.0.1.

OpenHIP and OpenVPN proof of concept

The proof of concept test was a success. After both machines had started their OpenHIP software and started the OpenVPN with their separate configurations, a point-to-point tunnel was successfully created with the LSI of Computer(B) as remote address. This forces Computer(A) to use HIP, which was successfully negotiated and set up. As proof Computer(A) is able to log in over SSH to Computer(B) over the newly created point-to-point tunnel on the configured address 10.8.0.1. This is shown in Figures 27 and 28.

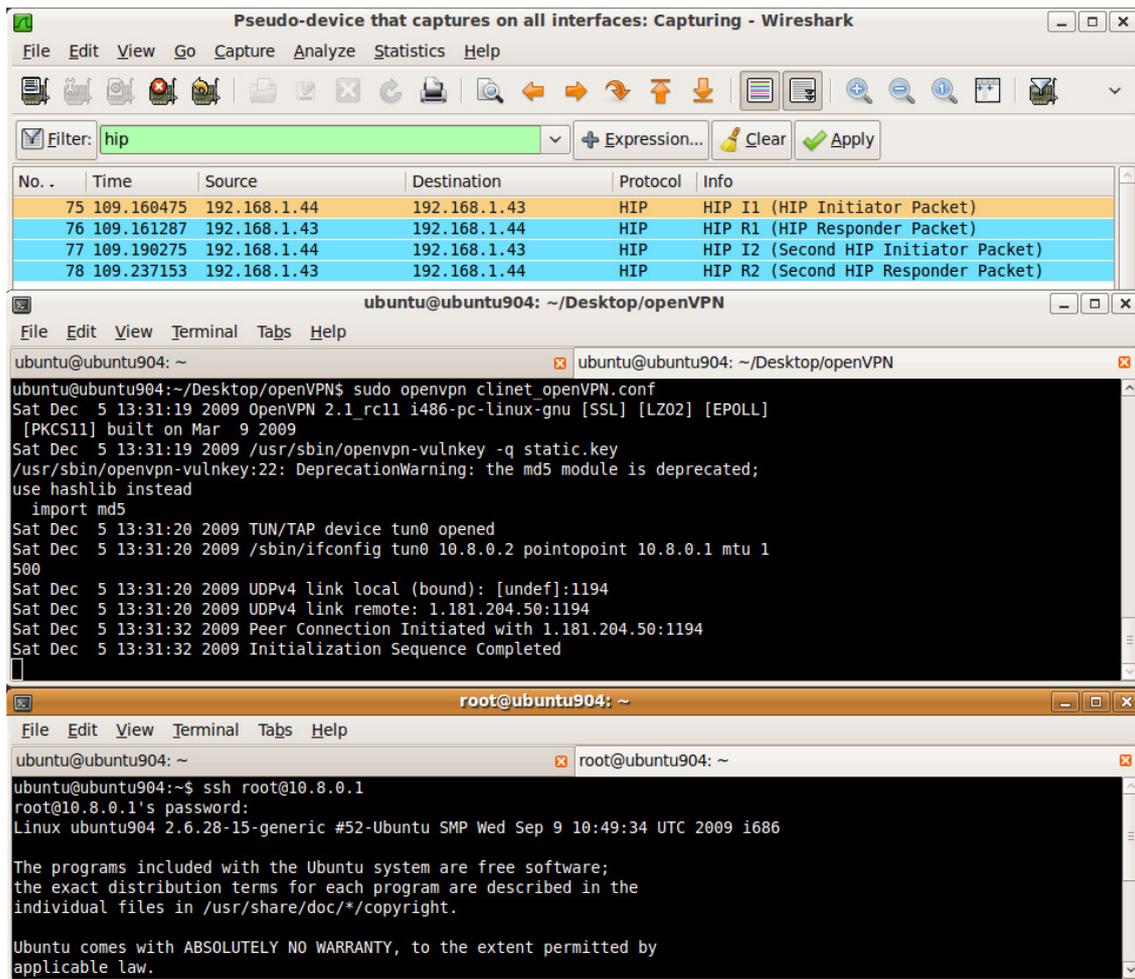


Figure 27. A point-to-point connection is successfully set up over HIP. Computer(A) is able to login to the server over SSH and the created point-to-point connection..

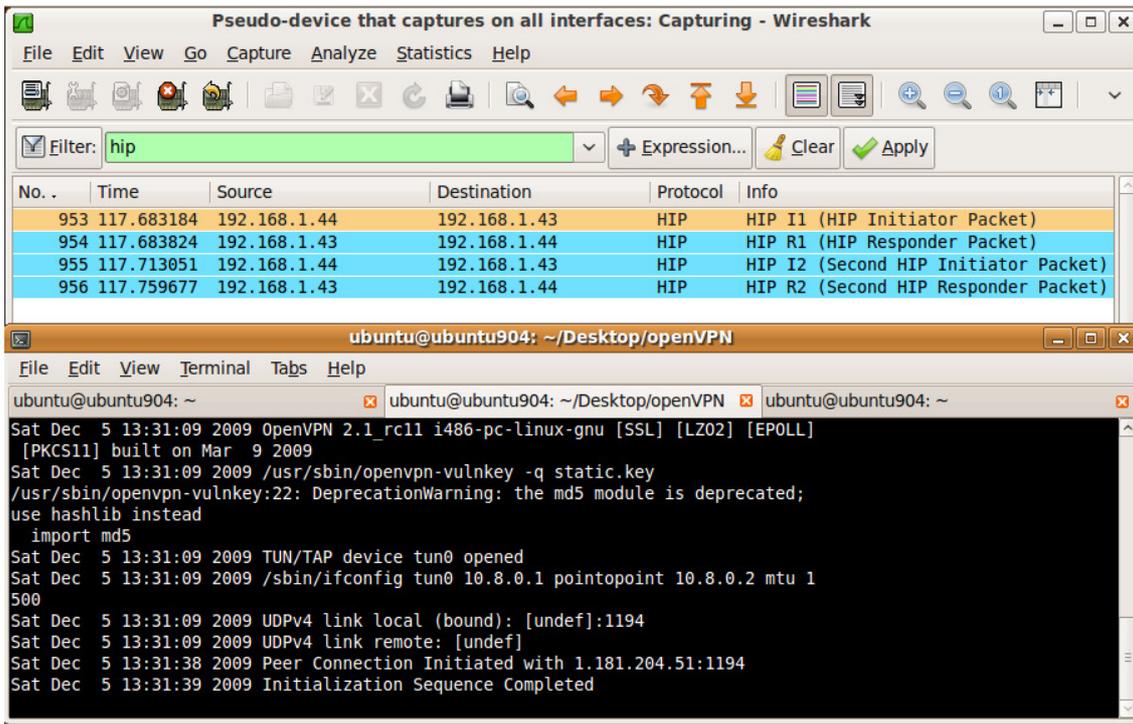


Figure 28. The server, Computer(B), of the test. The figure shows that OpenVPN sets up the connection using the locally configured LSI (1.181.204.51) of Computer(A). This is a proof that the connection is set up over HIP.

OpenHIP and OpenVPN - mobility

The mobility test starts in the same way as the proof of concept test. However, after the connection is set up, Computer(A) randomly changes the state of own network interfaces between “up” and “down”. The expected result is that the VPN connection stays alive while the HIP connection will be updated or re-negotiate.

Figure 29 depicts the client side, Computer(A) is in a make-before-break scenario.

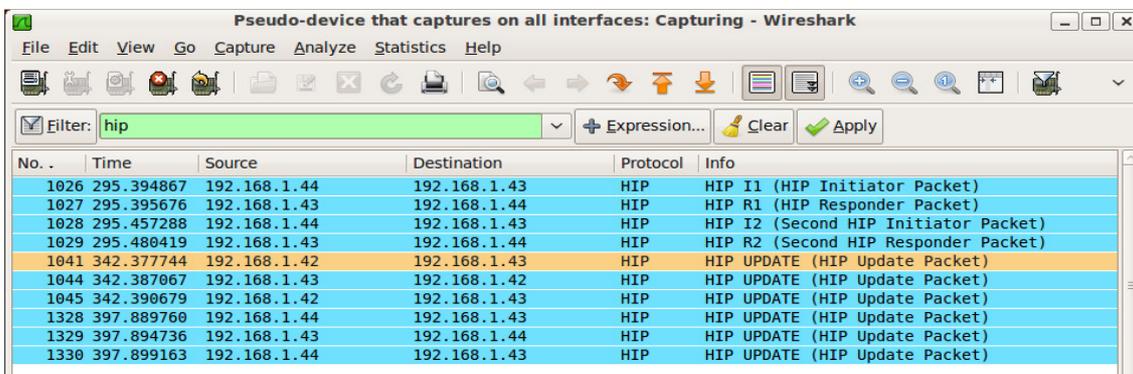


Figure 29. A series of successful updates sent from the client, Computer(A), to the server, Computer(B).

Figure 30 depicts the server side, Computer(B), in a make-before-break scenario.

No. .	Time	Source	Destination	Protocol	Info
80	271.362298	192.168.1.44	192.168.1.43	HIP	HIP I1 (HIP Initiator Packet)
81	271.362892	192.168.1.43	192.168.1.44	HIP	HIP R1 (HIP Responder Packet)
82	271.424720	192.168.1.44	192.168.1.43	HIP	HIP I2 (Second HIP Initiator Packet)
83	271.447583	192.168.1.43	192.168.1.44	HIP	HIP R2 (Second HIP Responder Packet)
97	318.345268	192.168.1.42	192.168.1.43	HIP	HIP UPDATE (HIP Update Packet)
100	318.354251	192.168.1.43	192.168.1.42	HIP	HIP UPDATE (HIP Update Packet)
101	318.358099	192.168.1.42	192.168.1.43	HIP	HIP UPDATE (HIP Update Packet)
351	373.867446	192.168.1.44	192.168.1.43	HIP	HIP UPDATE (HIP Update Packet)
352	373.872110	192.168.1.43	192.168.1.44	HIP	HIP UPDATE (HIP Update Packet)
353	373.876801	192.168.1.44	192.168.1.43	HIP	HIP UPDATE (HIP Update Packet)

Figure 30. A series of successful updates sent from the client, Computer(A), to the server, Computer(B).

Figure 31 shows that in a break-before-make scenario the client, Computer(A), tries to change the HIP connection to the point-to-point tunnel interface causing the communication to break and to remain unestablished.

No. .	Time	Source	Destination	Protocol	Info
1044	342.387067	192.168.1.43	192.168.1.42	HIP	HIP UPDATE (HIP Update Packet)
1045	342.390679	192.168.1.42	192.168.1.43	HIP	HIP UPDATE (HIP Update Packet)
1328	397.889760	192.168.1.44	192.168.1.43	HIP	HIP UPDATE (HIP Update Packet)
1329	397.894736	192.168.1.43	192.168.1.44	HIP	HIP UPDATE (HIP Update Packet)
1330	397.899163	192.168.1.44	192.168.1.43	HIP	HIP UPDATE (HIP Update Packet)
4181	842.835746	10.8.0.2	192.168.1.43	HIP	HIP UPDATE (HIP Update Packet)
4258	853.857284	10.8.0.2	192.168.1.43	HIP	HIP UPDATE (HIP Update Packet)
4382	864.769403	10.8.0.2	192.168.1.43	HIP	HIP UPDATE (HIP Update Packet)
4462	875.786700	10.8.0.2	192.168.1.43	HIP	HIP UPDATE (HIP Update Packet)
4509	883.757881	10.8.0.2	192.168.1.43	HIP	HIP I1 (HIP Initiator Packet)
4550	894.770514	10.8.0.2	192.168.1.43	HIP	HIP I1 (HIP Initiator Packet)
4575	905.762531	10.8.0.2	192.168.1.43	HIP	HIP I1 (HIP Initiator Packet)
4602	916.771538	10.8.0.2	192.168.1.43	HIP	HIP I1 (HIP Initiator Packet)

Figure 31. The client side tries to use the point-to-point interface as its communication link.

Verdict, OpenHIP and OpenVPN in combination is a promising combination and it works great in make-before-break scenarios. However, in scenarios where the connection is lost OpenHIP tries to use the only “active” network interface available which is the point-to-point interface causing the connection to break and renders it non-renegotiable for as long as the point-to-point is available. Even if the other interfaces states are set to “up” OpenHIP does not change its preferred address from the point-to-

point address to these. If it would be possible to exclude interfaces in the configuration of OpenHIP, this would not be a problem.

5.2.4 OpenHIP and SSH

There are two types of tests conducted in this section. The first test is the proof of concept and the second test will determine mobility and maturity. In this test OpenHIP and SSH will be used to mount a network drive to one computer from another. The OpenHIP SSH combination is expected to work together and provide easy to use Mobile VPN for the end user.

Setup consists of two computers running Linux, Ubuntu 9.04 and OpenHIP 0.7 binary build. The tools for mounting a network drive over SSH is a built-in feature in Gnome which comes as standard on Ubuntu.

The first test is to test the concept of the protocol and program combination. This is done without mobility being tested. The second stage tests the mobility if the first stage is successful.

Configuration of the OpenHIP known hosts file has been done before the tests are run. This configuration is made so that the computers can find each other since no DNS or rendezvous functionality is used.

OpenHIP and SSH – proof of concept

Computer(A) will attempt to mount the home folder of Computer(B) by using the HIP of Computer(B) as address for the SSH connection. The test is expected to run the HIP Base Exchange when SSH is called with the HIT of Computer(B) and before SSH negotiation. After HIP Base Exchange and SSH negotiation is done it is expected that the mounted network drive, the home folder of Computer(B), will be visible and function as a drive unit on Computer(A).

Figures 32 and 33 depict that proof of concept testing was successful.

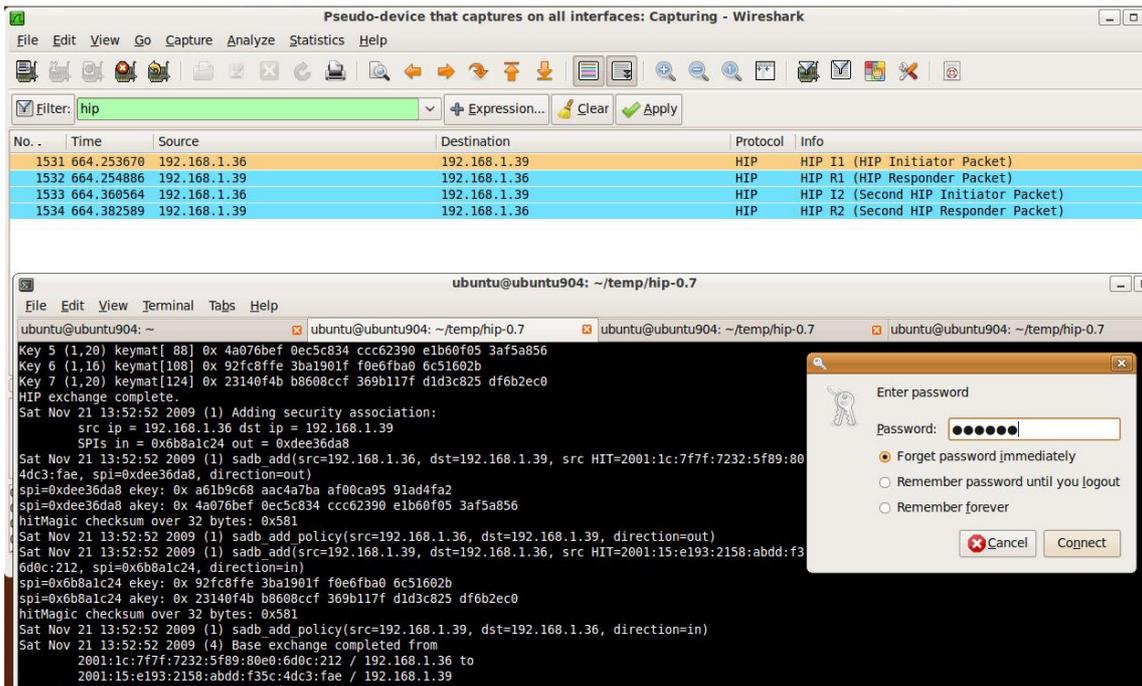


Figure 32. The HIP Base Exchange is done before SSH negotiation, initiator.

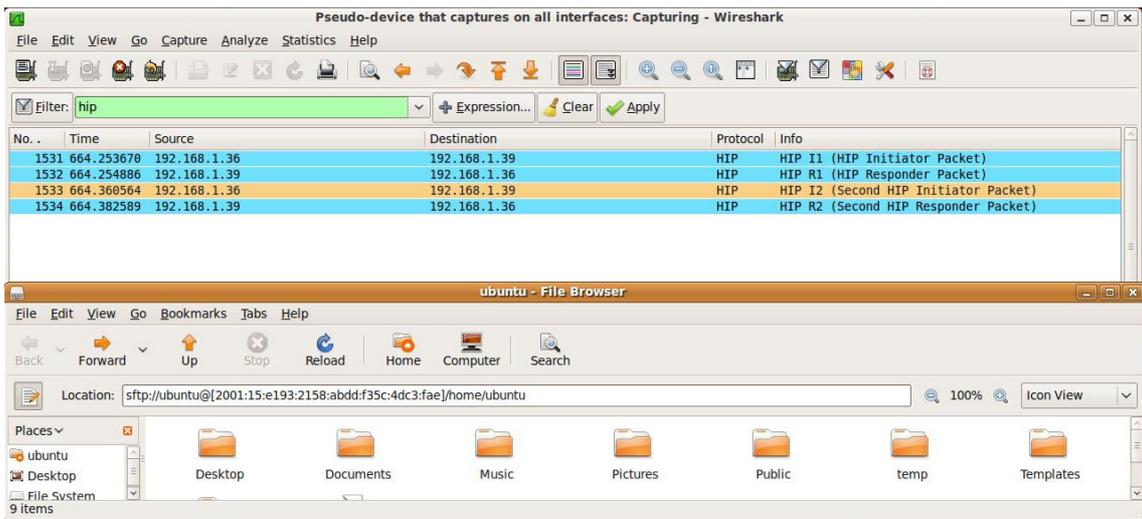


Figure 33. The mounted network drive is ready for use. As seen in the address bar it is using the peer HIT as address.

OpenHIP and SSH – mobility

Since the proof of concept testing is successful, the protocol and program setup can proceed to mobility testing. Mobility is tested by changing the states of the network interface of Computer(A). It is expected that the interface states can be changed randomly in both make-before-break and break-before-make scenarios. It is also expected that it will not be necessary for the OpenHIP software to do any re-negotiation

in the form of new Base Exchanges. Rather HIP update is expected to provide the necessary information to Computer(A) and Computer(B). If re-negotiation is needed the mobility is broken and session persistence is not achieved.

The combination of OpenHIP and SSH performed very well in the mobility tests. All make-before-break scenarios worked as expected. However break-before-make did not work and re-negotiation by the HIP software occurred. Thereby the session persistence was lost, but after the renegotiation the communication worked as well as before.

Figures 34, 35 and 36 show successful HIP updates and that the mounted network drive stays connected and operational.

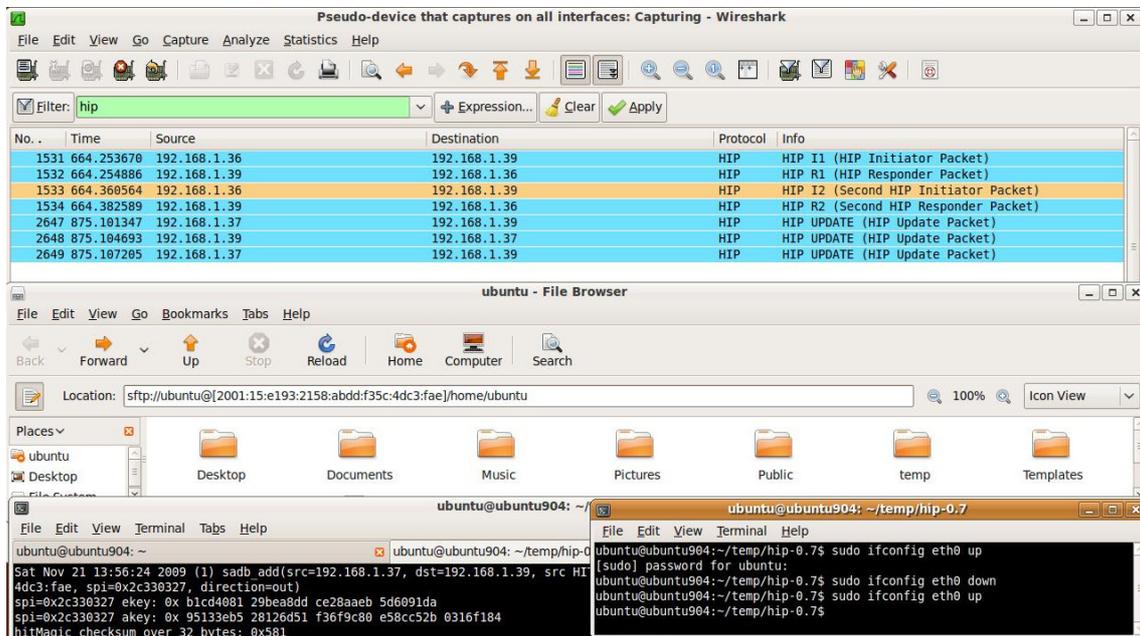


Figure 34. Make-before-break HIP update and the mounted network drive stays operational, (Computer(A)).

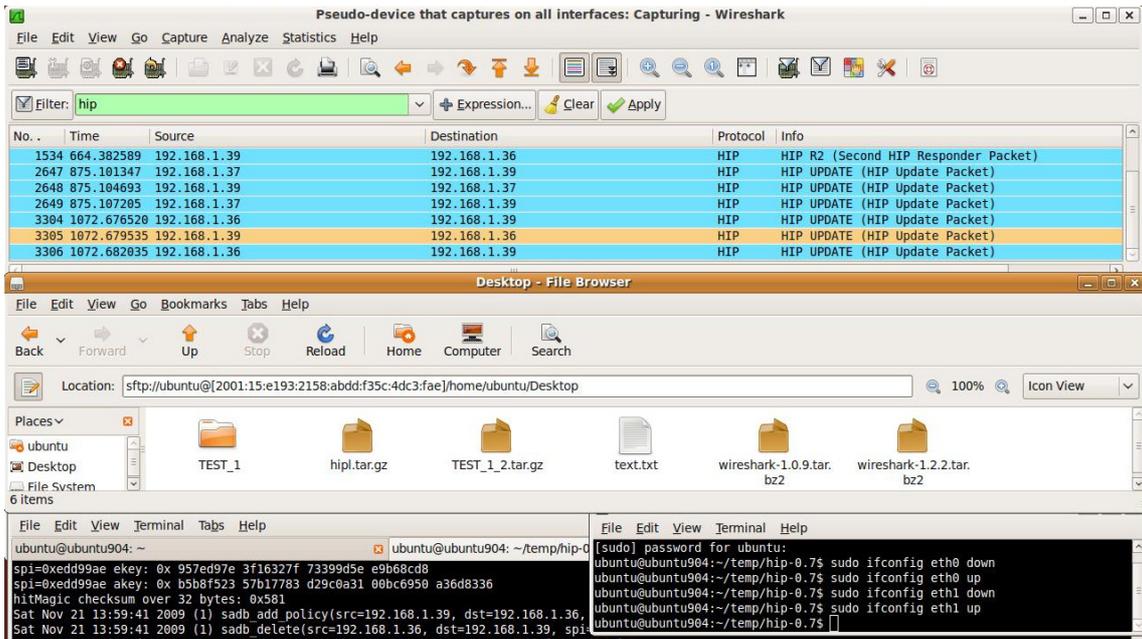


Figure 35. Make-before-break HIP-update and network drive stays operational, (Computer(A)).

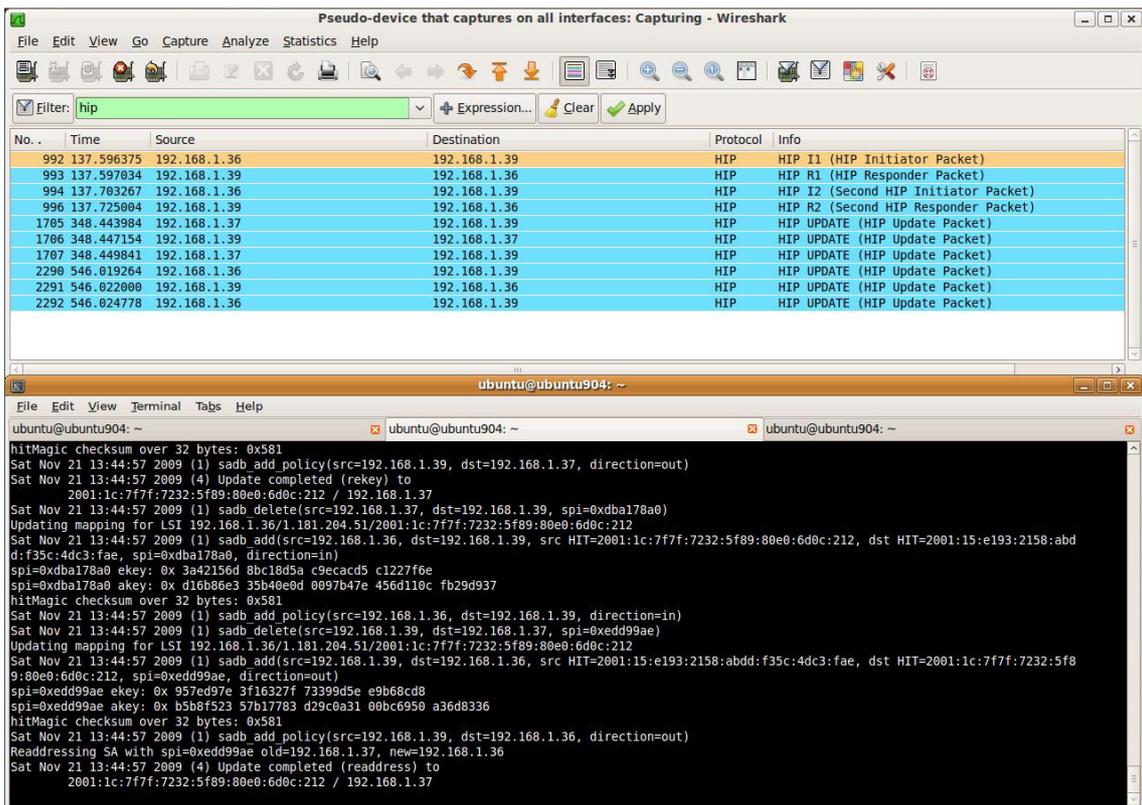


Figure 36. HIP update messages being received and processed by the responder (Computer(B)), no re-negotiation needed.

Figures 37, 38 and 39 show how the break-before-make scenario did not work and how Computer(A) and Computer(B) had to re-negotiate their HIP SAs. Both network

interfaces are down in Figure 37 and Computer(A) tries to use own IPv6 localhost for interface. In Figure 38 network interface eth0 of Computer(A) is turned on, and HIP update messages sent by Computer(A) are refused by Computer(B). This leads to HIP re-negotiation through a new HIP Base Exchange. Figure 39 shows Computer(B) refusing the HIP-updates and the re-negotiation.

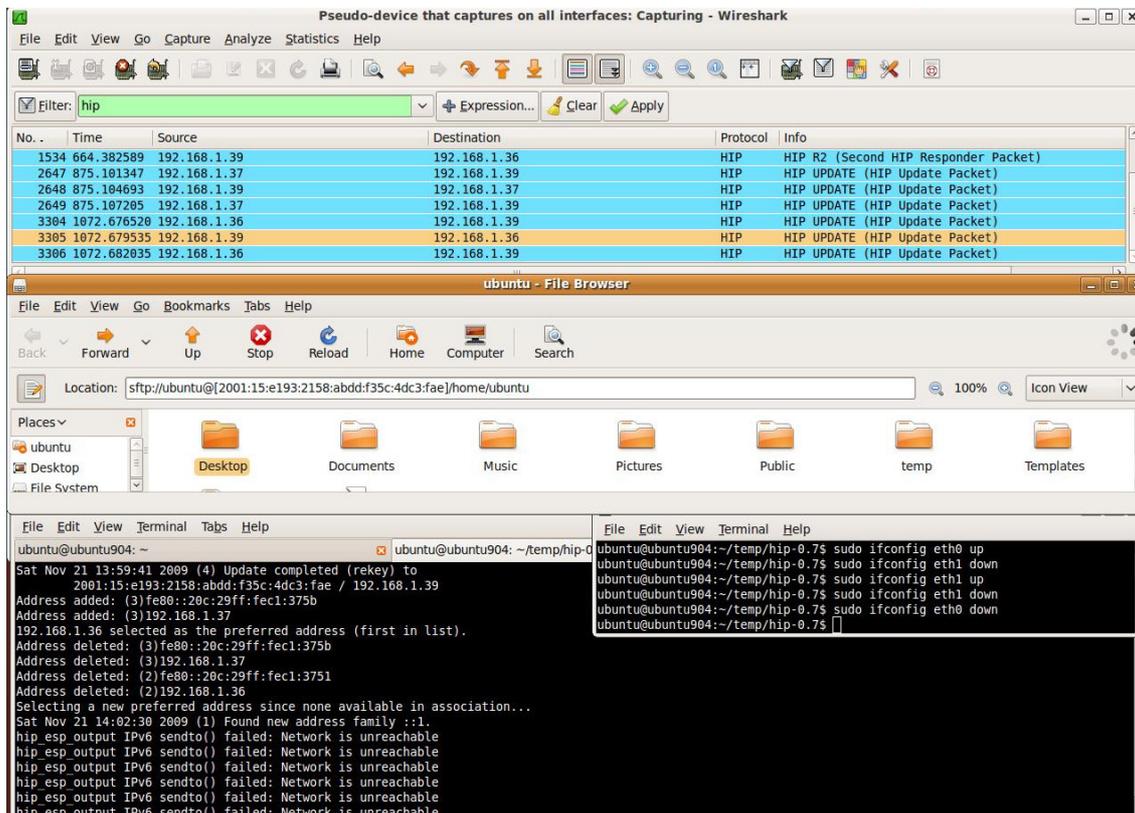


Figure 37. Computer(A) HIP update messages are impossible since there is no working interface. Also the mounted network is not operational at this point.

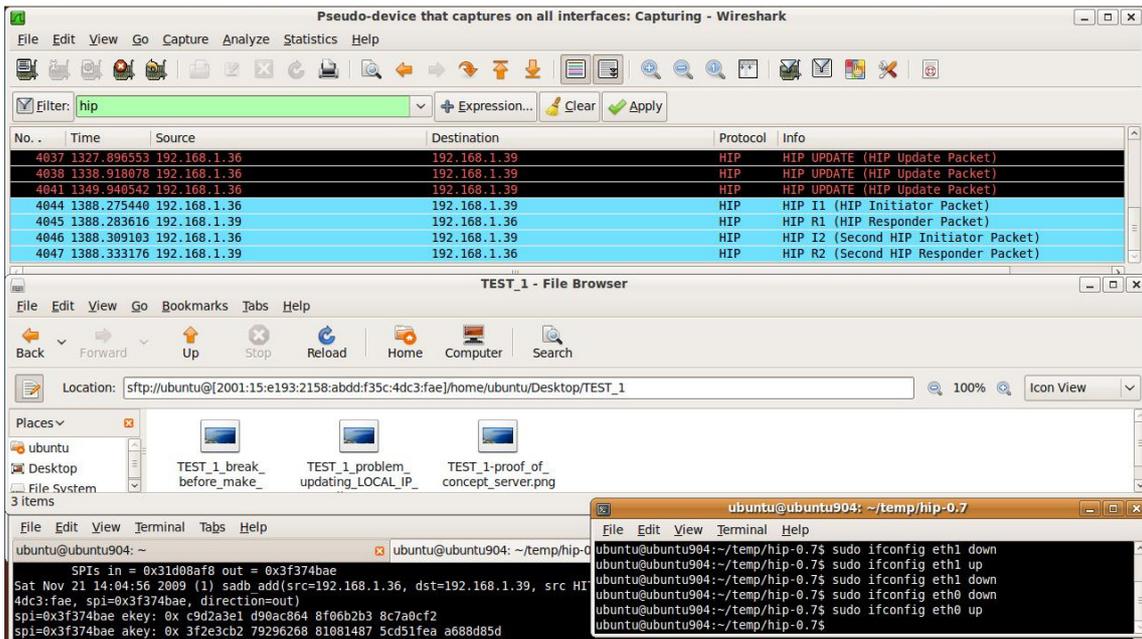


Figure 38. The HIP SA is re-negotiated after HIP update messages failed. The mounted network drive is again operational.

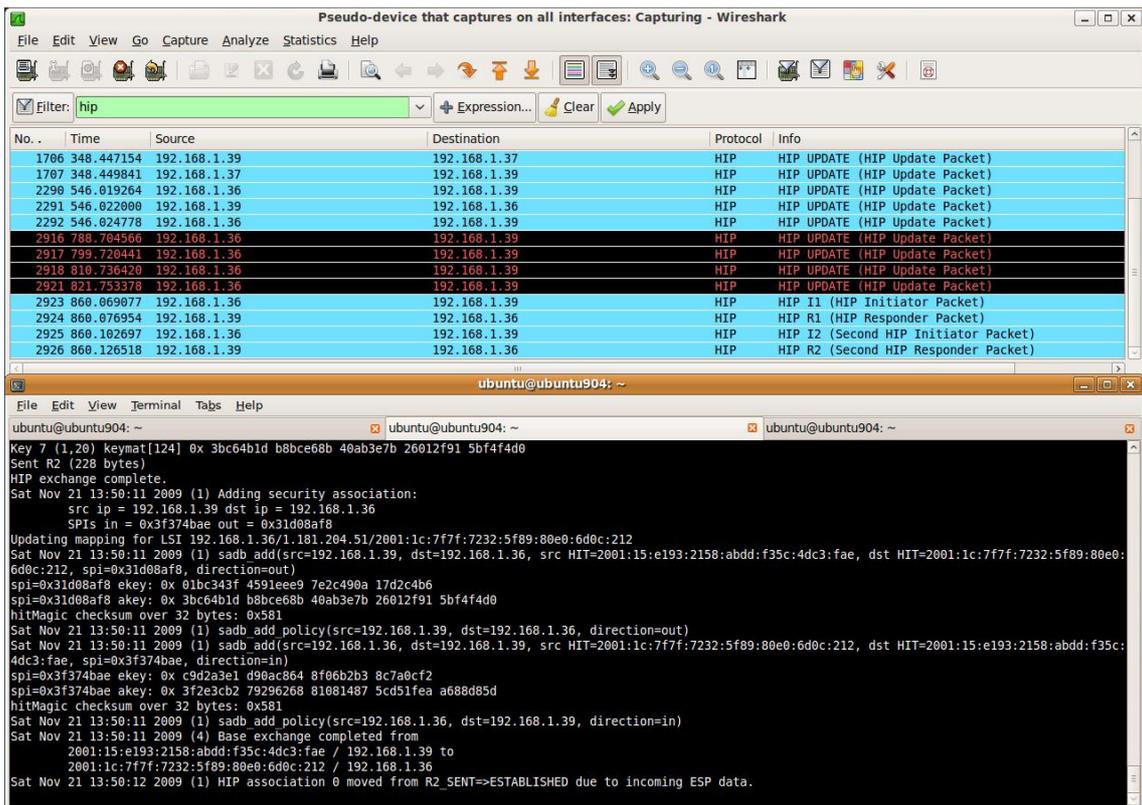


Figure 39. Computer(B) refusing HIP update messages and re-negotiates a HIP SA.

The OpenHIP and SSH combination was a successful combination and the same functionality can be expected from OpenHIP Windows Client. Still it was not perfect since the break-before-make did not work.

Verdict, this combination and OpenHIP in general seems to be very stable. Even if it didn't pass all expected tests, it still performed well and predictably.

5.2.5 Ocala – HIP proxy

The Ocala software is supposed to provide a generic proxy for which different plugins providing different protocols can be used. However, the Ocala –project has not been active since 2006 and because of that the testing has been extremely complicated.

Since Ocala and the HIP plug-in demand various libraries and a custom kernel, it is hard to produce a working test environment including a working Ocala proxy and a HIP plugin. The reason is that the Linux kernel has been updated many times since Ocala was developed. Ocala and the HIP plugin are therefore not compatible with kernels of present date. A custom built kernel from about 2006 compiled with the custom modules provided by OpenHIP or InfraHIP is required.

No tests have yet been successful. Therefore testing with Ocala remains undone. However, the theory behind the Ocala proxy is good, and if the project would be still running it would probably be one of the better solutions for providing Mobile VPN.

5.3 Practical conclusion

The practical part has reviewed several different solutions on how to provide Mobile VPN. However none of the solutions using the experimental protocol HIP was 100% successful. This proves that HIP is not yet ready to be introduced as a Mobile VPN solution for common use. However, HIP has great performance and low overhead when it is working. HIP can be recommended for professionals and enthusiasts. Figure 40 shows the different conclusions for the tests done.

f = full support
 p = partial support (mostly successful)
 n = none, or most tests failed

Protocol implementation	OpenVPN		SSH hdd mapping	
	concept	mobility	concept	mobility
OpenHIP	f	p	f	f
InfraHIP	f	n	f	p

Figure 40. The success of the different HIP implementations and different programs.

The evaluation of other protocols and programs has been strictly theoretical, reviewing features and architectural design. These solutions have been based on MIP. Programs using MIP for Mobile VPN are already commercially available, and companies such as Cisco have developed their own Mobile VPN. Even though MIP is a developed protocol improvements and new ideas are still under development for the protocol and for the use of the protocol. The SUM architecture is described earlier is a good example of this.

Present solutions using MIP or MIP and MOBIKE are far more successful for common use of Mobile IP than solutions based on HIP. The reason is the immaturity of HIP.

6 DISCUSSION

This Bc.S. thesis has examined the mobility protocols HIP and MIP as well as the security protocols IPsec, SSL/TLS and SSH. Also MOBIKE has been examined.

MOBIKE ties together the mobility protocols and IPsec protocol. An evaluation of the protocols supports the understanding of the practical part of the thesis. However, the practical part has been focused mostly on implementations using HIP. The reason is my involvement with the WISEciti project in ARCADA with its focus on HIP. HIP is a mobility protocol which is not yet implemented in commercial VPN solutions.

Therefore no complete solution is offered yet. The practical part has examined two different HIP implementations OpenHIP and InfraHIP. These have been identically tested with different software to determine the best combination of protocol implementation and VPN software. The tests showed that OpenHIP, which is developed and used by Boeing, was much more mature than the rival InfraHIP. InfraHIP, which is developed by HIIT, is presently used for testing and for scientific experiments.

The practical part has been a bit changed from the original plan. The plan was to test as many protocols and VPN solutions as possible. This turned out to be harder and more time consuming than first expected. Also, limitations on the hardware available during the thesis project have been an issue. Furthermore, the practical part has been quite successful in determining the usability and maturity of both available HIP implementations and how these implementations can be used with present programs, that offer different features and levels of security. Even though the SSH solution is not a real VPN solution, I think it shows how HIP can easily provide security and mobility to ordinary users.

To conclude this Bc.S. thesis, the plan has been followed in terms of content but not in terms of the time schedule. The part which differs most from the plan is the practical part. This was also the part that I paid the least attention to during the plan, since I thought it would sort itself out. In my opinion, the thesis been successful, but it has over time evolved in a different direction than the plan originally outlined.

LIST OF REFERENCES

MOBIKE

P. Eronen, Ed. 2006. IKEv2 Mobility and Multihoming Protocol (MOBIKE), RFC 4555, IETF. Retrieved April 25, 2009. <http://www.ietf.org/rfc/rfc4555.txt>

T. Kivinen, H. Tschofenig. 2006. Design of the IKEv2 Mobility and Multihoming (MOBIKE) Protocol, RFC 4621, IETF. Retrieved April 25, 2009. <http://www.ietf.org/rfc/rfc4621.txt>

IPsec

Frankel S., Kent K., Lewkowski R., Orebaugh A., Ritchey R., Sharma S. 2005. Guide to IPsec VPNs, Recommendations of the National Institute of Standards and Technology, NIST. Retrieved March 25, 2009. <http://csrc.nist.gov/publications/nistpubs/800-77/sp800-77.pdf>.

Kent S., Seo K. 2005. Security Architecture for the Internet Protocol, RFC 4301, IETF. Retrieved March 24, 2009. <http://tools.ietf.org/html/rfc4301#section-3.1>.

Kent S. 2005. IP Authentication Header, RFC 4302, IETF. Retrieved March 31, 2009. <http://tools.ietf.org/html/rfc4302>

Kent S. 2005. IP Encapsulating Security Payload (ESP), RFC 4303, IETF. Retrieved March 31, 2009. <http://tools.ietf.org/html/rfc4303>

C. Kaufman, Ed. 2005. Internet Key Exchange (IKEv2) Protocol, RFC 4306, IETF. Retrieved March 31, 2009. <http://tools.ietf.org/html/rfc4306>

SSL/TLS

T. Dierks, E. Rescorla. 2008. The Transport Layer Security (TLS) Protocol Version 1.2, RFC 5246, IETF. Retrieved April 28, 2009. <http://tools.ietf.org/html/rfc5246#section-6>

G.Magnini. 2005. Introduction to SSL. Retrieved April 28, 2009. https://developer.mozilla.org/en/Introduction_to_SSL

SSH

T. Ylonen C, Lonvick Ed. The Secure Shell (SSH) Protocol Architecture, RFC 4251, IETF. Retrieved January 17, 2010. <http://tools.ietf.org/html/rfc4251>

SSH Filesystem. Retrieved January 17, 2010. <http://fuse.sourceforge.net/sshfs.html>

GNOME 2.22 Release Notes, 2008. Retrieved January 17, 2010. <http://library.gnome.org/misc/release-notes/2.22/#sect:gvfs-gio>

Mobile IP

Charles E. Perkins. 1998. Mobile Networking Through Mobile IP, IEEE Internet Computing, vol. 2. Retrieved March 18, 2009. IEEE xplore, <http://ieeexplore.ieee.org/stampPDF/getPDF.jsp?tp=&arnumber=656077&isnumber=14306>.

Charles E. Perkins. Original 1997, 2002. Mobile IP, IEEE Communications Magazine – 50th Anniversary Commemorative Issue/May 2002. Retrieved March 18, 2009. http://www.ee.oulu.fi/~skidi/teaching/mobile_and_ubiquitous_multimedia_2002/Perkins.pdf

Johnson, D., Perkins, C., Arkko, J. 2004. Mobility Support in IPv6, IETF, RFC 3775. Retrieved March 24, 2009. <http://www.ietf.org/rfc/rfc3775.txt>.

Johnson, D., Perkins, C., Arkko, J. 2009. Mobility Support in IPv6 draft-ietf-mext-rfc3775bis-03.txt, IETF, DRAFT. Retrieved March 24, 2009. <http://www.ietf.org/rfc/rfc3775.txt>.

HIP

R. Moskowitz & P. Nikander. 2008. Host Identity Protocol (HIP) Architecture. MEMO. Retrieved March 1, 2009. <http://www.ietf.org/rfc/rfc4423.txt>

InfraHIP Project: Intro, 2009. Retrieved March 1, 2009. <http://infrachip.hiit.fi/index.php?index=how>

HIP for BSD Project: Documentation, HIP Implementation for FreeBSD (PDF). 2005. Retrieved March 1, 2009. <http://hip4inter.net/documentation/hip4bsd.pdf>

Pekka Nikander, 2004. HIPpy Road Warriors Jumping Hoods over Road Blocks. Retrieved March 11, 2009. http://hiprg.piuha.net/workshop/nikander_road_warrior.pdf

Patrik Salmela and Jan Malén. 2006. E-business and Telecommunication Networks, Host Identity Protocol Proxy. Retrieved March 12, 2009

Practical studies

Virtual Machines, Virtual Server, Virtual Infrastructure – VMware: Virtualization Basics. 2009. Retrieved April 28, 2009. <http://www.vmware.com/technology/virtual-machine.html>

Wireshark: About. 2009. Retrieved April 30, 2009. <http://www.wireshark.org/about.html>

The Ubuntu Story | Ubuntu. 2009. Retrieved April 30, 2009. <http://www.ubuntu.com/community/ubuntustory>

What is Ubuntu? | Ubuntu. 2009. Retrieved April 30, 2009. <http://www.ubuntu.com/products/whatisubuntu>

InfraHIP Project: About. 2009. Retrieved April 30, 2009. <http://infrachip.hiit.fi/index.php?index=about>

InfraHIP Project: HIPL. 2009. Retrieved April 30, 2009. <http://infrachip.hiit.fi/index.php?index=hipl>

OpenVPN Compatibility. 2009. Retrieved May 1, 2009. <http://infrachip.hiit.fi/hipl/manual/ch16s07.html>

About OpenVPN. 2009. Retrieved May 1, 2009. <http://openvpn.net/index.php/home/55-about-openvpn.html>

Cisco Mobile VPN - Enabling Cisco End-Device Based IP Mobility. 2006. Retrieved December 5, 2009.

http://www.ciscosystems.org.ro/en/US/prod/collateral/iosswrel/ps6537/ps6551/ps6744/prod_white_paper0900aecd803a837c.pdf

Cisco Mobile IP. 2001. Retrieved January 12, 2010.

http://www.cisco.com/warp/public/cc/pd/iosw/prodlit/mbxul_wp.pdf

Birdstep Technology Oy, 2009. SagfeMove, Secure Seamless Mobility. Retrieved January 18 2009.

http://www.nomasis.ch/fileadmin/user_upload/flyer/produkte/birdstep/productsheet_safemove_5.0.pdf

APPENDIX A: MOBIKE - MN CHANGES POINT OF ATTACHMENT

This example illustrates how a MN changes its point of access and updates the change to the IPsec gateway (RFC 4555. 2006).

Step 1 is the normal IKE_INIT exchange;

```
Initiator                               Responder
-----                               -
(IP_I1:500 -> IP_R1:500)
HDR, SAi1, KEi, Ni,
  N(NAT_DETECTION_SOURCE_IP),
  N(NAT_DETECTION_DESTINATION_IP)  -->

<-- (IP_R1:500 -> IP_I1:500)
HDR, SAr1, KEr, Nr,
  N(NAT_DETECTION_SOURCE_IP),
  N(NAT_DETECTION_DESTINATION_IP)
```

Step 2, the peers inform each other that they support MOBIKE;

```
Initiator                               Responder
-----                               -
(IP_I1:4500 -> IP_R1:4500)
HDR, SK { IDi, CERT, AUTH,
  CP(CFG_REQUEST),
  SAi2, TSi, TSr,
  N(MOBIKE_SUPPORTED) }  -->

<-- (IP_R1:4500 -> IP_I1:4500)
HDR, SK { IDr, CERT, AUTH,
  CP(CFG_REPLY),
  SAr2, TSi, TSr,
  N(MOBIKE_SUPPORTED) }
```

(Initiator gets information from lower layers that its attachment point and IP address have changed.)

Step 3, the initiator notices a change in its own IP address and informs the responder about this by sending an INFORMATIONAL request containing the UPDATE_SA_ADDRESSES notification. The request is sent using the new IP address. At this point, it also starts to use the new address as a source address in its own outgoing ESP traffic. Upon receiving the UPDATE_SA_ADDRESSES notification, the responder records the new address and, if it is required by policy, performs a return routability check of the IP address;

```

Initiator                               Responder
-----                               -
IP_I2:4500 -> IP_R1:4500)
  HDR, SK { N(UPDATE_SA_ADDRESSES),
            N(NAT_DETECTION_SOURCE_IP),
            N(NAT_DETECTION_DESTINATION_IP) } -->

                                <-- (IP_R1:4500 -> IP_I2:4500)
                                HDR, SK { N(NAT_DETECTION_SOURCE_IP),
                                N(NAT_DETECTION_DESTINATION_IP) }

```

(Responder verifies that the initiator has given it a correct IP address.)

Step 4, when the routability check is completed the responder starts to use the new address as the destination for its outgoing ESP traffic;

```

Initiator                               Responder
-----                               -
                                <-- (IP_R1:4500 -> IP_I2:4500)
                                HDR, SK { N(COOKIE2) }

(IP_I2:4500 -> IP_R1:4500)
HDR, SK { N(COOKIE2) } -->

```

APPENDIX B: COMPILATION GUIDE FOR WIRESHARK WITH HIP PATCH

This guide is for Ubuntu/Linux and does not take any other distributions or operating systems into consideration. This guide will step by step show how to get the source needed and how to patch it and compile it to a working version of Wireshark with support for HIP.

Step 1, Download the source needed, Wireshark v1.0.7 from <http://media-2.cacotech.com/wireshark/src/wireshark-1.0.7.tar.bz2>. and the needed patch from InfraHIP homepage; <http://hipl.hiit.fi/hipl/hipl.tar.gz>

Step 2, Get the build essentials for Wireshark by executing the command `sudo apt-get build-dep wireshark`.

Step 3, Patch Wireshark, locate the patch in the source from infrahip, in the folder `patches/ethereal/`. The patches used are `wireshark-1.1.0-hip.patch` and `wireshark-1.1.0-hip-midauth.patch`.

Step 4, Go to the Wireshark folder that contains the `autogen.sh` and execute the following commands;

```
patch -p1 < <PATH-TO-PATCH>/wireshark-1.1.0-hip.patch
patch -p1 < <PATH-TO-PATCH>/wireshark-1.1.0-hip-midauth.patch
./autogen.sh
./configure
make
sudo make install
```

Step 5, This should compile without problem and a HIP supporting version of Wireshark is created. To see that HIP actually is supported, start Wireshark, go to `Edit->Preference->supported protocols` and browse until HIP is found. Alternatively run an application using HIP and see that it recognizes it.

If there is any problem with patching or compiling, please refer to the individual project documentation.

APPENDIX C: OPENVPN CONFIGURATION FILES AND SETUP

The `static.key` is a static key to be used as a pre-shared secret between the server and the client. The key is generated with the command: `openvpn --genkey --secret static.key`. Since the static key is a shared secret it has to be shared and not generated individually for the server and the client.

The configuration provided creates a virtual network interface for both the client and the server. The VPN tunnel is created with the server end-point `10.8.0.1` and the client at `10.8.0.2`. The encrypted communication will be created with UDP and the port used is `1194`. The remote variable in the client configuration is IP address or the domain name of the server. The server is passive and waits for a client to connect. If the shared secret is correct a secure VPN tunnel is created.

The OpenVPN server and client have to be started with root privileges, in Ubuntu; `sudo openvpn --config openVPN_client.conf` for the client and `openvpn --config openVPN_server.conf` for the server.

Configuration for InfraHIP and OpenVPN

Client configuration

```
#openVPN_client.conf

#Here we are using the locally added LSI for our server running HIP
remote 10.0.0.50

dev tun
ifconfig 10.8.0.2 10.8.0.1
secret static.key
```

Server configuration

```
#openVPN_server.conf
port 1194
dev tun
ifconfig 10.8.0.1 10.8.0.2
secret static.key
```

Configuration files for OpenHIP and OpenVPN

Client configuration

```
#openVPN_client.conf

port 1194
#Here we are using the locally added LSI for our server running HIP

Remote 1.181.204.50
dev tun
ifconfig 10.8.0.2 10.8.0.1
secret static.key
```

Server configuration

```
#openVPN_server.conf
port 1194
dev tun

ifconfig 10.8.0.1 10.8.0.2
secret static.key
```