Evaluating IP Security and Mobility on Lightweight Hardware

Licentiate Thesis

Andrey Khurri

Supervisor: Professor Antti Ylä-Jääski

Instructor: Adjunct Professor Andrei Gurtov

Evaluators: Dr. Bengt Ahlgren and Professor Klaus Wehrle

Helsinki University of Technology
Faculty of Information and Natural Sciences
Department of Computer Science and Engineering

ABSTRACT OF LICENTIATE THESIS	HELSINKI UNIVERSITY OF TECHNOLOGY P. O. BOX 1000, FI-02015 TKK http://www.tkk.fi

Author Andrey Khurri

Name of the thesis
Evaluating IP Security and Mobility on Lightweight Hardware

Manuscript submitted 22.05.2009	Manuscript revised 23.08.2009
Date of the presentation 03.06.2009	
[X] Monograph	[ ] Article dissertation (summary + original articles)

Faculty	Faculty of Information and Natural Sciences
Department	Department of Computer Science and Engineering
Field of research	Telecommunications Software
Evaluator (s)	Dr. Bengt Ahlgren and Professor Klaus Wehrle
Supervisor	Professor Antti Ylä-Jääski
Instructor	Adjunct Professor Andrei Gurtov

Abstract

This work presents an empirical evaluation of applicability of selected existing IP security and mobility mechanisms to lightweight mobile devices and network components with limited resources and capabilities. In particular, we consider the Host Identity Protocol (HIP), recently specified by the IETF for achieving authentication, secure mobility and multihoming, data protection and prevention of several types of attacks. HIP uses the Diffie-Hellman protocol to establish a shared secret for two hosts, digital signatures to provide integrity of control plane and IPsec ESP encryption to protect user data. These computationally expensive operations might easily stress CPU, memory and battery resources of a lightweight client, as well as negatively affect data throughput and latency.

We describe our porting experience with HIP on an embedded Linux PDA, a Symbian-based smartphone and two OpenWrt Wi-Fi access routers, thereby contributing to the protocol deployment. We present a set of measurement results of different HIP operations on these devices and evaluate the impact of public-key cryptography on the processor load, memory usage and battery lifetime, as well as the influence of the IPsec encryption on Round-Trip Time and TCP throughput. In addition, we assess how the lightweight hardware of a mobile handheld or a Wi-Fi access router in turn affects the duration of certain protocol operations including HIP base exchange, HIP mobility update, puzzle solving procedure and generation of an asymmetric key pair. After analyzing the empirical results we make conclusions and recommendations on applicability of unmodified HIP and IPsec to resource-constrained mobile devices. We also survey related work and draw parallels with our own research results.

Keywords IP, security, mobility, performance, lightweight hardware

ISBN (printed)	ISSN (printed)
ISBN (pdf)	ISSN (pdf)
Language English	Number of pages 20 + 96 p.
Publisher	
Print distribution	

[ ] The thesis can be read at http://lib.tkk.fi/Diss

Acknowledgements

This thesis is the result of my three-year work at Helsinki Institute for Information Technology and postgraduate studies at Helsinki University of Technology. The work presents accumulated research on evaluating applicability of existing IP security and mobility mechanisms to resource-constrained mobile devices, such as a Linux-based Nokia Internet Tablet, a Symbian-based smartphone and a Wi-Fi access router running OpenWrt.

The results and findings presented in this thesis are partly an outcome of our joint work with several people at HIIT, TKK and partner organizations. Without you it would have been hard to achieve many goals. I would like to express my deepest gratitude to my supervisor Prof. Antti Ylä-Jääski and my instructor Adj. Prof. Dr. Andrei Gurtov. Andrei originally invited me to work at HIIT, motivated me to eventually start my postgraduate studies and supported me all along this thorny path. Thank you, Andrei, for giving me this opportunity, for your valuable feedback and support. I am very grateful to Prof. Antti Ylä-Jääski from TKK for supervising this thesis, providing useful advice, comments and specific guidelines concerning my postgraduate study. Thank you, Antti, for your assistance and always being supportive.

The work has been done within two projects, *MERCoNe* (Multiaccess Experimentation in Real Converging Networks, 2006-2008) and *WISEciti* (Wireless Community Services for Mobile Citizens, 2008-2010). I would like to thank the funding organizations of these projects including TEKES and several industrial partners. My particular thank-you goes to all the individual researchers working with whom in both projects has been a great and invaluable experience. I am grateful to our international partners from the Distributed Systems Group at RWTH Aachen University with whom we have had several fruitful discussions and points of collaboration. My special word of thanks is to Tobias Heer, Klaus Wehrle and René Hummen.

I wish to personally thank numerous researchers from the Networking Research Group at HIIT who have generously spared their time to help me in understanding and resolving non-trivial issues, discussing research (and not only) ideas and debugging the code. Miika Komu, Andrey Lukyanenko, Dmitriy Kuptsov, Boris Nechaev, Kristiina Karvonen, Joakim Koskela, Oleg Ponomarev, Samu Varjonen, Juho Heikkila, Theofanis Kilinkaridis and Tatiana Polishchuk, my sincere thanks go to all of you. Besides, I would like to thank the rest of my colleagues from HIIT, especially those whom I do not mention explicitly here. Working with you has been

a great pleasure. Prof. Martti Mäntylä and Dr. Pekka Nikander are the ones who were the source of inspiration during our rare but fruitful meetings. Thank you Pirkko, Päivi, Assel, Tuomo and Andrea for your continuous assistance with all administrative tasks, for your smiles and positive mood.

I am thankful to all of my friends in Finland and in Russia for being generally curious about my work and studies. Your questions to me have often been a chance to evaluate the essence of my research activities from another perspective. Thank you for your constant friendship.

My parents, Nadezda and Alexander, and my sister, Natalia, though remotely, have always been with me supporting each of my steps throughout these days in Finland. I am grateful to you for growing me up, for your endless parental care, your love and generous help. Thank you for supporting and encouraging me in all my activities. I owe you a lot.

Finally, I devote most of my thanks, words of appreciation and love to one particular person, Ekaterina. You have become the closest and the dearest person to me. Thank you infinitely for your true love and tender care. I am very happy with you.

Espoo, May 2009

Contents

List of Abbreviations

ACL Access Control List

AES Advanced Encryption Standard

AH Authentication Header

AR Access Router

API Application Programming Interface

ARP Address Resolution Protocol

BE Base Exchange

BEET Bound End-to-End Tunnel

BSD Berkeley Software Distribution

CA Certificate Authority

CBA Credit-Based Authorization

CPU Central Processing Unit

DH Diffie-Hellman

DHCP Dynamic Host Configuration Protocol

DHT Distributed Hash Table

DoS Denial of Service

DSA Digital Signature Algorithm

ESP Encapsulated Security Payload

FQDN Fully Qualified Domain Name

FTP File Transfer Protocol

GNU GNU is not Unix

GPRS General Packet Radio Service

GUI Graphical User Interface

HIP Host Identity Protocol

HIPD HIP Daemon

HIPL HIP for Linux

HIT Host Identity Tag

HMAC Hash Message Authentication Code

HTTP Hyper Text Transfer Protocol

IMEI International Mobile Equipment Identity

ICMP Internet Control Message Protocol

IP Internet Protocol

IPsec IP security

ID Identifier

IEEE Institute of Electrical and Electronics Engineers

IPR Intellectual Property Rights

IETF Internet Engineering Task Force

IKE Internet Key Exchange

IPv6 Internet Protocol version 6

IRTF Internet Research Task Force

LAN Local Area Network

LHIP Lightweight HIP

MIP Mobile IP

MOBIKE Mobile Key Exchange

MR Mobile Router

NAT Network Address Translator

NEMO Network Mobility

OS Operating System

OSI Open Systems Interconnection

OSS Open Source Software

P2P Peer-to-peer

PC Personal Computer

PDA Personal Digital Assistant

PISA P2P Internet Sharing Architecture

POSIX Portable Operating System Interface

QoS Quality of Service

RFC Request for Comments

RFID Radio-frequency Identification

RSA Rivest-Shamir-Adleman algorithm

RTT Round-Trip Time

SA Security Association

SADB Security Association Database

SIP Session Initiation Protocol

SPI Security Parameter Index

SSH Secure Shell

SDK Software Development Kit

TCP Transmission Control Protocol

UDP User Datagram Protocol

UMTS Universal Mobile Telecommunication System

URI Universal Resource Identifier

VoIP Voice over IP

VPN Virtual Private Network

Wi-Fi Wireless Fidelity

WLAN Wireless Local Area Network

WPA Wi-Fi Protected Access

List of Figures

List of Tables

Author's contribution

The results presented in this thesis have been published in three scientific articles [54, 53, 61]. This section identifies the author's contribution in these research papers and other relevant activities.

The article *Performance of Host Identity Protocol on Lightweight Hardware*[1] [54] was the first publication on evaluating feasibility of running IP-based security and mobility solutions, such as HIP, on resource-constrained mobile devices. I ported the existing HIPL implementation to Nokia 770 Internet Tablet, an embedded Linux PDA. This task was followed by a set of extensive protocol and network measurements performed jointly with Ekaterina Vorobyeva. The results were then analyzed by all the co-authors. I wrote the majority of the manuscript parts including the background chapter on HIP, the chapter describing the porting process, and the chapter with measurement results and analysis. I participated in writing the introduction and conclusion, as well as visualizing the results with the *gnuplot* tool. I presented the final version of the article at the *Second ACM/IEEE International Workshop on Mobility in the Evolving Internet Architecture* in Kyoto, Japan in August 2007.

The article *Performance of Host Identity Protocol on Symbian OS*[2] [53] continued addressing the same topic while focusing on another operating system and another class of mobile devices. The paper described migration of two separate HIP implementations, *HIPL* and *OpenHIP*, to Symbian OS and presented measurement results conducted in a pilot network with Nokia E51 and N80 smartphones. It took a big effort to eventually run the HIPL software (originally written for Linux) on a Symbian device. With the help of two summer interns and the assistance of other colleagues, I accomplished porting of the HIPL base protocol engine, though leaving the IPsec support unimplemented for a number of reasons. Especially debugging the protocol on the Nokia smartphones required considerable effort. The OpenHIP implementation was ported and measured by Dmitriy Kuptsov. Together with him, we performed HIP experimentations and collected measurement results that were then analyzed by all the co-authors. I was the primary writer of the article's

[1] A. Khurri, E. Vorobyeva, and A. Gurtov. 2007. Performance of Host Identity Protocol on Lightweight Hardware. In: Proc. of the 2nd ACM/IEEE International Workshop on Mobility in the Evolving Internet Architecture (MobiArch'07). ACM, New York, NY, USA. ISBN 978-1-59593-784-8.

[2] A. Khurri, D. Kuptsov, and A. Gurtov. 2009. Performance of Host Identity Protocol on Symbian OS. In: Proc. of the IEEE International Conference on Communications 2009 (ICC'09). IEEE.

text except for few of its parts such as the OpenHIP description and the section about RAM usage. I presented the article at the IEEE International Conference on Communication in Dresden, Germany in June 2009.

In the article *Distributed User Authentication in Wireless LANs*[3] [61] I contributed to the architecture design, the experimental network setup and the analysis of the measurement results on La Fonera FON2100 and Gateworks Avila GW2348-4 wireless ARs. I provided the graphs presenting two different authentication architectures, as well as enhanced other figures illustrating measurement results. I actively participated in the writing process by providing the whole background chapter on HIP and the sections describing the experimental testbed and panOULU deployment. In addition, I reviewed the whole manuscript and contributed text to its other parts.

[3]D. Kuptsov, A. Khurri, and A. Gurtov. 2009. Distributed User Authentication in Wireless LANs. In: Proc. of the 10th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'09). IEEE.

1 Introduction

In this thesis we present an evaluation of applicability of existing IP security and mobility mechanisms for lightweight mobile devices with limited resources and capabilities. The following sections introduce to the reader the nature of the research problem, specify our contribution and describe the organization of the thesis.

1.1 IP technology goes embedded

Migration of mobile telecommunication systems to IP technology is a growing, well recognized trend. The existing mobile devices such as PDAs and smartphones are increasingly utilizing the TCP/IP communication stack to interconnect between each other and transfer data. However, the traditional Internet has donated to its mobile counterpart not only a set of technologies but also a number of research issues associated with the efficient use of these technologies. Moreover, the scale of the problems in the new wireless and mobile environment, which has different properties and requirements, has dramatically increased. In particular, such environment, compared to wired and stationary communications systems, is a) more prone to the attacks and b) requires that terminals are mobile. This in turn denotes two main essential attributes for a device operating in the mobile Internet: security and mobility, or secure mobility.

A natural way to address these issues would be to borrow the existing solutions that have proved to work on conventional computers. As an example, there are several projects that successfully miniaturized the TCP/IP stack and made it perfectly suited for the micro-systems with only few hundreds bytes of available RAM [19, 17]. However, these solutions have been so far concentrating on the communications performance and general ability to run IP on tiny objects rather than on reliable security and mobility support. Mobility has not been in focus during the design of the original Internet, whereas modern network-layer security mechanisms often rely on cryptography, which involves computationally-intensive operations that might stress the resource-constrained devices.

Within the context of this thesis, we call "resource-constrained", "limited", "restricted" or "lightweight" such communication devices that are mobile and battery-powered and have limited computational resources compared to conventional laptops and desktop computers. Such devices are PDAs, cellular phones, Internet tablets, communicators etc. The target mobile devices in this work have the CPU clock

rate ranging from 220 to 369 MHz, the amount of RAM ranging from 64 to 96 MB and the battery capacity ranging from 860 to 1500 mAh. As an additional class of "lightweight" devices in this thesis, we experiment with small network components such as Wi-Fi access routers with a 183-MHz CPU and 16 MB of RAM.

1.2 TCP/IP challenges on lightweight mobile devices

The TCP/IP communication stack provides full functionality for interconnecting mobile devices and transferring data between them. However, initially constructed for stationary computers several decades ago, today TCP/IP technologies have little or no capabilities to cope with the new issues introduced by the mobile Internet and the lightweight portable devices with limited hardware capabilities.

A well known problem with the existing TCP/IP stack running on mobile devices is that being borrowed from the traditional stationary Internet it does not provide means for seamless mobility when a client changes its network attachment point. This results in broken connections, delays, and inability to deliver services to moving entities.

Another crucial aspect of mobile devices is communication security, which is especially threatened in wireless networks vulnerable to a variety of attacks. Despite of many proposed security mechanisms, it is not obvious how efficient they are on lightweight mobile devices in the presence of limited computational and battery resources.

1.3 Thesis contribution

Our key contribution in this thesis is the evaluation of the applicability of IP-based security and mobility to lightweight mobile device such as cellular phones, Internet tablets and Wi-Fi access routers. In other words, our goal is to assess how feasible it is to run the existing security and mobility solutions on these classes of devices. Our interest is especially generated by the use of cryptography that is often a major part of security and mobility architecture intended to authenticate mobile Internet hosts, protect communications between them, and defend from potential attacks. Such operations, as encryption and public key signatures implemented in software, are computationally-intensive and could stress CPU and battery resources of mobile devices.

As a basis for our empirical evaluation we use the Host Identity Protocol (HIP), which is an experimental secure mobility protocol specified by the IETF [71, 73, 47, 63, 62, 80, 79]. HIP can be fairly considered as a universal solution to many Internet problems as it provides support of end-to-end security, resistance to CPU and memory exhausting denial-of-service (DoS) attacks, NAT traversal, mobility and multihoming. HIP uses the Diffie-Hellman protocol to create a session key for two communicating hosts and IPsec encapsulation for data packets.

A good property of HIP from the research and experimentation perspective is that it relies on public-key cryptography and IPsec. This makes HIP similar to other security and mobility protocols and, thus, provides an opportunity to correlate measurement results obtained in this work with other projects. On the other hand, as HIP builds its operations on cryptographic identities, its integration with other security protocols becomes easier [29]. With all its advantages, HIP is a prospective candidate to be an essential part of the future Internet architecture.

Although several previous projects evaluated HIP on standard Internet hosts [40, 38, 82, 48], none of the studies has targeted a HIP assessment on a mobile device with restricted resources. In this work, we perform HIP measurements on real mobile devices: Linux-based Internet tablets and Symbian-based smartphones. The extensive analysis of the results allows us to provide a set of recommendations on using unmodified HIP on lightweight clients in varying circumstances. Our experience with HIP reported in this thesis is of particular importance to the Internet research community. This is because HIP is still an experimental protocol and for large-scale deployment it needs more success stories and real-life pilot examples.

Another contribution of this work is a HIP-based distributed authentication architecture that addresses important security and mobility issues in wireless local area networks. To validate our model we build a pilot WLAN with two models of Wi-Fi access routers that have varying hardware resources. We evaluate performance of selected public-key operations on these Wi-Fi access routers. This assessment allows to determine a set of criteria for selecting equipment while deploying such an architecture in different networks.

We further contribute to the security and mobility issues on lightweight devices by surveying related work and drawing appropriate parallels with our own research results.

As an additional contribution, we consider our first-hand knowledge about migration of open source software (OSS) projects to mobile environment. In particular, we report our porting experience of two different HIP implementations to Symbian

OS, as well as migration of HIPL to an embedded Linux OS and to the OpenWrt platform.

1.4 Thesis structure

The structure of the thesis is organized as follows. In Chapter 2, we overview related technologies from the security and mobility fields. In addition, we present the background on the Host Identity Protocol that has been the target of our empirical research. Chapter 3 surveys the most interesting pieces of related work found in the literature. Chapter 4 and Chapter 5 describe the nature of the research problem and research methodology. In Chapter 6, we present performance evaluation of HIP on Nokia 770 Internet Tablet. The chapter contains description of our port of HIPL and protocol performance results on the mobile Internet tablet. In particular, our measurements include data throughput, latency, and power consumption of a HIP base exchange and a HIP mobility update. Chapter 7 describes our experience of running HIP on Symbian OS and evaluates feasibility of running IP-based security on lightweight cellular phones. We measure and analyze duration of the HIP base exchange and its parts, CPU load, RAM utilization and power consumption of several protocol operations on Nokia E51 and Nokia N80. The chapter elaborates on the porting process of two HIP implementations, *HIPL* and *OpenHIP*, to Symbian OS. Chapter 8 presents our authentication architecture in wireless LANs and reports performance results of its components running on two Wi-Fi AR models, La Fonera FON2100 and Gateworks Avila GW2348-4. We summarize the empirical results, discuss the outcome of our research and highlight potential future research directions in Chapter 9. Finally, Chapter 10 concludes the thesis.

2 Background

This chapter presents a brief overview of several related to our research concepts and definitions from the IP security and mobility fields. It does not intend to provide details but rather aims at helping the reader in understanding selected ideas, terms and techniques.

2.1 IP security

The aim of IP security in general is to protect network-layer communications between the hosts in the Internet. The main components of IP security include authentication, authorization, privacy, confidentiality, protection of data integrity, and prevention of a variety of attacks by means of the different protocols and mechanisms. In the following sections we give an overview of several security techniques, including *symmetric*, *public-key* and *elliptic curve* cryptography. Cryptography mechanisms in communications systems are used to ensure data integrity and confidentiality by encrypting the messages sent over an insecure channel and to authenticate their originator by signing the messages digitally. The encrypted pieces of the data are usually referred to as *ciphertext*, whilst the algorithms to perform encryption and decryption operations are known as *ciphers*. Ciphertexts themselves make little sense to an entity that is not able to decrypt them and, therefore, are usually safe to transmit them over an insecure network. Digital signatures, in turn, allow to verify the authenticity of a message originator to its receivers [14, 16, 89, 93]. Below we give a short overview of the most prominent cryptography algorithms and protocols consolidated in three different groups, *symmetric cryptography*, *public-key cryptography* and *elliptic curve cryptography*.

2.1.1 Symmetric cryptography

To perform encryption and decryption operations the sender and receiver of a message have to employ a key. In symmetric cryptography, either this key is identical for both encryption and decryption or derivation of one key from another is trivial. Thus, for confidential communication, the symmetric key must not be revealed to any third party. This makes the key management process very challenging. The main questions are how to securely select a cryptographic key, how to distribute it to both communicating parties and how to store the key safely on the hosts. However,

the complexity of managing the keys in symmetric cryptography is compensated by relatively low computational cost comparing to other encryption algorithms such as public-key cryptography. Two types of the ciphers utilized in symmetric cryptography are *stream* and *block* ciphers. Stream ciphers encrypt bits of a message one by one, whereas block ciphers divide the message on blocks of different size and operate on them [14]. Examples of the most known block cipher methods used with symmetric cryptography are DES (Data Encryption Standard) [74], AES (Advance Encryption Standard) [76], and Blowfish [91].

2.1.2 Public-key cryptography

Public-key cryptography also known as *asymmetric cryptography* uses an approach different from symmetric cryptography. The key idea is built around a pair of the keys, public and private, that complement each other and are generated together on a host. The public key is open to anyone and is distributed among the host's peers. The peers that want to communicate with the host use its public key to encrypt the messages. These messages can only be decrypted using the corresponding private key of a public-private key pair. Since the private key is kept secretly on the host possessing it, asymmetric cryptography provides high level of communication security and data protection. One of the main properties of asymmetric cryptography is that the private key cannot be practically derived from its public counterpart [16, 89].

Public-key cryptography allows not only to preserve communication privacy between a host and its peers by making it impossible to read transferred information by the third parties but can also be used to digitally sign messages. A public-key signature tightly binds a message with its originator and proves that a particular message has been signed by a particular host. Having signed a message its sender cannot repudiate its actions, i.e. refuse its involvement in creating the message. Practically this means that after the message had been signed the host cannot change the message content without modifying its signature and vice versa [89]. A downside of public key cryptography is that the algorithms it uses operate on large numbers, involve heavy mathematical calculations and, thus, produce significant cost to communicating hosts. This limits the applicability of the public-key algorithms, especially in cases when one or more of the communicating parties is a lightweight mobile host with constrained hardware resources. In practise, asymmetric and symmetric cryptography algorithms are often utilized jointly. First, public key cryptography is used to securely establish a shared secret between two hosts. Then, the hosts

can employ this shared secret in symmetric cryptography algorithms that are more efficient computationally [14].

RSA

RSA algorithm [89] received its name from the initial letters of surnames of its original inventors, Rivest, Shamir and Adleman. In 1978 they published an encryption method to achieve privacy and authentication in electronic communications systems, which later had become one of the commonly used cipher methods in public-key cryptography. With this method, constructed on the complexity of factoring large prime numbers, it is computationally difficult to derive the private part of a public-private key pair by knowing its public component. Thus, it becomes possible to eliminate the secrecy of the public key while distributing it over an insecure channel. Encryption of the messages with the public key of a host allows to protect integrity of transmitted data (that may include symmetric keys as well), while signing the messages with the private key authenticates the host to its peers. More details about the RSA algorithm can be found in [89].

DSA

DSA is an acronym for *Digital Signature Algorithm*, which is a standard published by the US National Institute of Standards and Technology (NIST) in 2000. As the name suggests, DSA can be used to digitally sign messages whilst it cannot be used to perform data encryption. The full standard specification can be found in [75].

Diffie-Hellman

Diffie-Hellman (DH) [16] is a key exchange protocol introduced by Whitfield Diffie and Martin E. Hellman in 1976. The protocol is an example of the common use of the public-key and symmetric cryptography algorithms to solve the key distribution problem in the cryptosystems. The DH protocol can be used together with RSA and DSA algorithms to securely exchange the keys of two communicating hosts and eventually generate a pairwise secret. This secret is subsequently used to derive a common session key used in actual data communication between these hosts [16].

Table 2.1: Comparable key sizes with different cryptosystems. Adapted from [13].

Symmetric	**ECC**	**DSA/RSA**	**"Best Before"**
80	160	1024	2010
112	224	2048	2030
128	256	3072	2040
192	384	7680	2080
256	512	15360	2120

2.1.3 Elliptic Curve Cryptography

An approach to use elliptic curves in cryptography as an alternative to the existing public-key algorithms (in particular, the Diffie-Hellman protocol) has been proposed by Victor Miller in 1985 [70]. The main author's emphasis was that Elliptic Curve Cryptography (ECC) could achieve more efficient computation comparing to the DH protocol. Neal Koblitz is another famous scientist well known for his valuable contribution to the ECC field (e.g., [55, 56]). Along with Miller, he is referred to as one of the ECC's originators. The strength of the ECC idea lies on the elliptic curve discrete logarithm problem (ECDLP), which in this case is much harder to solve computationally than with the ECC's counterparts. Largely for this reason, ECC is often being called as a "new generation" of public-key cryptography. Although with present requirements to security levels it does not bring much efficiency comparing to the "mainstream" algorithms such as DH or RSA, ECC is believed to provide significant benefits for future communications systems when the security threats will be multiplied from their current level. One particular advantage of ECC is that it requires notably shorter key length than RSA and DH while claiming to preserve an equivalent security level (see Table 2.1). In other words, computation cost per bit is rapidly decreasing for ECC with increasing the public key length. Presently used RSA/DSA cryptosystems still often employ the key length of 1024 bits, which is considered to remain "unbreakable" until 2010 [13].

Looking at the current ECC implementation and deployment status one can observe serious implications. The biggest obstacle is that the majority of ECC-related techniques and aspects have been patented by numerous individuals and companies,

which substantially differentiates ECC from the rest of the cryptography methods. One notable player in this area is a Canadian company *Certicom* that holds over 100 patents concerned to ECC and public-key cryptography in general [77]. According to Certicom web site[4], since the introduction of ECC in 1985 the company has been continuously seeking to develop an efficient implementation of ECC and break a common belief in its relative slowness. As a result, Certicom has delivered a commercial ECC toolkit that can be used in numerous applications. Other interesting initiatives of Certicom include sponsoring of the Center for Advanced Cryptographic Research (CACR) at the University of Waterloo along with several annual ECC scientific venues for the world's key cryptographers and organizing since 1997 the Certicom ECC Challenge[5], which gives practically to anyone an opportunity to solve the ECDLP at its current level. Participants of the challenge are supposed to derive the ECC private keys based on the list of the public keys and other known parameters. As of now, the highest solved challenges are 109-bit. The 131-bit challenge requires much more resources to be solved whereas higher challenges (starting from 163-bit) are treated as computationally infeasible.

Besides mentioned activities, Certicom has formed the Standards for Efficient Cryptography Group (SECG)[6] that combines core providers of cryptography solutions seeking for interoperability between them. Among other, the SECG has been producing a number of ECC standards and documents including "SEC 1: Elliptic Curve Cryptography" [13] and "SEC 2: Recommended Elliptic Curve Domain Parameters" [12]. Despite of complex IPR issues around ECC, several countries around the world are adopting an extent of it via licensing. For instance, the National Security Agency (NSA) has purchased from Certicom the rights to use a set of ECC techniques under patents believing in their strong future potential. The US Department of Defense (DoD) is targeting to replace around 1.3 million existing hardware devices within a time frame of ten years. The goal is to ensure that future communication systems will be capable of protecting sensitive information for the US government and country [77]. Section 3.1 of this thesis surveys some relevant studies on ECC security on mobile devices.

[4] `http://www.certicom.com/index.php/ecc`
[5] `http://www.certicom.com/index.php/the-certicom-ecc-challenge`
[6] `http://www.secg.org`

2.2 IP mobility

Mobility is one of the essential attributes of today's Internet users and networks. Especially, the owners of mobile lightweight devices, such as cell phones and laptops, tend to change their geographical location that often causes changes of their network attachment point and their IP addresses. Other possible scenarios include switching application sessions between different hardware terminals and moving the whole networks at once. Mobility in general can be classified into different types depending on various factors, for example, which of the aforementioned changes are needed to be sustained, or which layer of the OSI reference model is in focus. Most of the authors working in the area of mobility distinguish between *terminal, network, flow, session, personal,* and *service mobility* [59, 67, 92, 107].

2.2.1 Mobility types

Terminal or *host mobility* implies uninterrupted TCP/IP connections in the presence of node mobility, i.e. upon changing its physical location and/or IP address [67, 92, 107]. *Network mobility* provides means to sustain moving of a complete network with its mobile access router and maintain reachability to all of the mobile nodes inside the network [15, 67, 107]. *Flow mobility* allows splitting parts of a connection between distinct interfaces of a host, as well as switching a communication between IPv4 and IPv6 protocols [107]. *Session mobility* denotes an ability to seamlessly move versatile communication sessions from one terminal to another, e.g., from a PDA to a PC, or from a laptop to a smartphone [92]. *Personal mobility* is defined as a possibility to make the user reachable via a single user identity (e.g., an URI, an IP address, a cryptographic key) on several physical appliances or several user identities on a single device, either at the same time or alternatively [92]. Finally, *service mobility* is referred to as a means to access personal user services in versatile circumstances, including changes of client devices and network access providers [92].

2.2.2 Handover types

A comprehensive set of mobility related definitions can be found in [67]. For the purpose of this thesis, in addition to the above mobility categories, it is necessary to provide a general definition of a handover. A *handover* or a *handoff* is a process of changing the network attachment point by a host, or an attempt to perform such a change. The aim of the most IP mobility mechanisms is to provide functionality that would allow to minimize session breaks and interruptions during a handoff [67].

When a change of the network attachment point is unnoticed by the user, the respective handover is often referred to as *seamless*. Handovers might differ by the scope (e.g., horizontal or vertical) and by the level of control (e.g., mobile-controlled or network-controlled). *Horizontal* handoff defines the process of switching the network attachment point of a mobile node between access points of the same type (e.g., WLAN to WLAN). In contrast, during a *vertical* handover a mobile device moves from one access network type to another, such as from WLAN to 3G. These and other definitions of handoff types can be found in [67].

2.2.3 Mobility at different layers

Different aspects of mobility have been extensively studied to date resulting in a wide range of mobility protocols and extensions that aim to operate on different, from link to application, layers of the OSI reference model [10, 46, 57, 59, 79, 84, 88, 97, 106, 107]. Sometimes, the distinction of the mobility mechanisms between different protocol stack layers is blurred. For instance, network (IP) layer mobility can be made transparent to upper layers, whereas network-layer applications can be assisted by application-layer mobility mechanisms. Resulting from this tendency, a number of combined protocol schemes for multi-layered mobility management have been proposed to date [95, 101, 102].

In this work, as the name suggests, we mostly consider IP-layer mobility that, in terms of the presented definitions, best refers to *terminal* or *host* mobility. In Chapter 3, we review related work in the area of IP mobility that has been following different approaches including *locator/identifier split*. Section 2.3 below provides background information on the Host Identity Protocol, which follows the latter approach and has been the key technology in the experimental part of this thesis.

2.3 Host Identity Protocol

The existing Internet architecture that had been primarily designed for stationary hosts nowadays faces many non-trivial challenges with the growing amount of mobile terminals. Currently, there are two name spaces that are used globally by the Internet services and applications, domain names and IP addresses. IP addresses serve the dual role in the Internet being both end host identifiers and topological locators. This general principle does not allow hosts to change their location without breaking ongoing transport protocol connections that are strictly bound to IP addresses.

2.3.1 HIP architecture

The Host Identity Protocol (HIP) [47, 62, 63, 71, 73, 79, 80] had been proposed to overcome the above mentioned problem. The idea behind HIP is decoupling of the network layer from the higher layers in the protocol stack architecture (see Figure 2.1). HIP defines a new global name space, the Host Identity name space, thereby splitting the double meaning of the IP addresses. When HIP is used, upper layers do not any more rely on IP addresses as host names. Instead, Host Identifiers are used in the transport protocol headers for identifying hosts and establishing connections. IP addresses at the same time act purely as locators and are responsible for routing packets towards the destination. A Host Identifier is a public key of the host. For compatibility with IPv6 legacy applications and to ease protocol implementations, a Host Identifier is further represented by a 128-bit long cryptographic hash, the Host Identity Tag (HIT). Each HIP-enabled host has one or more host identifiers that might be either public (available from a directory service) or unpublished (local).

HIP offers several benefits including end-to-end security, resistance to CPU and memory exhausting denial-of-service (DoS) attacks, NAT traversal, mobility and multihoming support.

2.3.2 Base exchange

To start communicating through HIP, two entities must establish a HIP association. This process is known as the HIP Base Exchange (BEX) [71, 73] and it consists of four messages transferred between the initiator and the responder. After BEX is successfully completed, both hosts are confident that private keys corresponding to Host Identifiers (public keys) are indeed possessed by their peers. Another purpose of the HIP base exchange is to create a pair of IPsec Encapsulated Security Payload (ESP) Security Associations (SAs), one for each direction. All subsequent traffic between communicating parts is protected by IPsec. A new IPsec ESP mode, Bound End-to-end Tunnel (BEET) [81] is used in HIP. The main advantage of BEET mode is low overhead in contrast to the regular tunnel mode.

Figure 2.1 illustrates the overall HIP architecture including the BEX. The initiator can retrieve the HI/HIT of the responder from a DNS directory [80] by sending a FQDN in a DNS query. Instead of resolving the FQDN to an IP address, the DNS server replies with an HI (FQDN→HI). Transport layer creates a packet with the HI as the destination point identifier. During the next step, HI is mapped to an

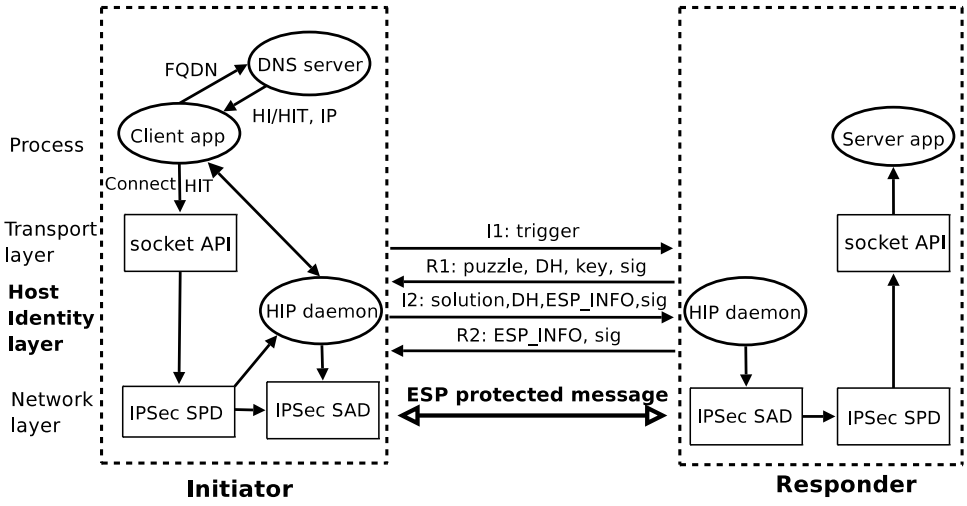


Figure 2.1: HIP architecture.

IP address by the HIP daemon on the Host Identity layer. Finally, the packet is processed by the network layer and delivered to the responder. As a result, the conventional 5-tuple socket becomes {protocol, source HI, source port, destination HI, destination port}.

2.3.3 Mobility and multihoming

Since neither transport layer connections nor security associations (SAs) created after the HIP base exchange are bound to IP addresses, a mobile client can change its IP address (upon moving, due to a DHCP lease or IPv6 router advertisement) and keep on transmitting ESP-protected packets to its peer. HIP supports such mobility events by implementing an end-to-end signaling mechanism between communicating nodes (see Figure 2.2) [79].

The purpose of the first UPDATE packet is to notify the peer of a new IP address and ESP information associated with this address. The corresponding parameters are called LOCATOR and ESP_INFO. The message also contains a SEQ parameter (a sequence number of the packet) and is therefore protected against possible losses by retransmission. Upon receiving the UPDATE message, the peer host must validate it, update any local HI↔IP mappings and assure that the mobile client is accessible via the new link. This is accomplished by sending the second UPDATE packet back to the mobile host at its new IP address containing an echo request along with the

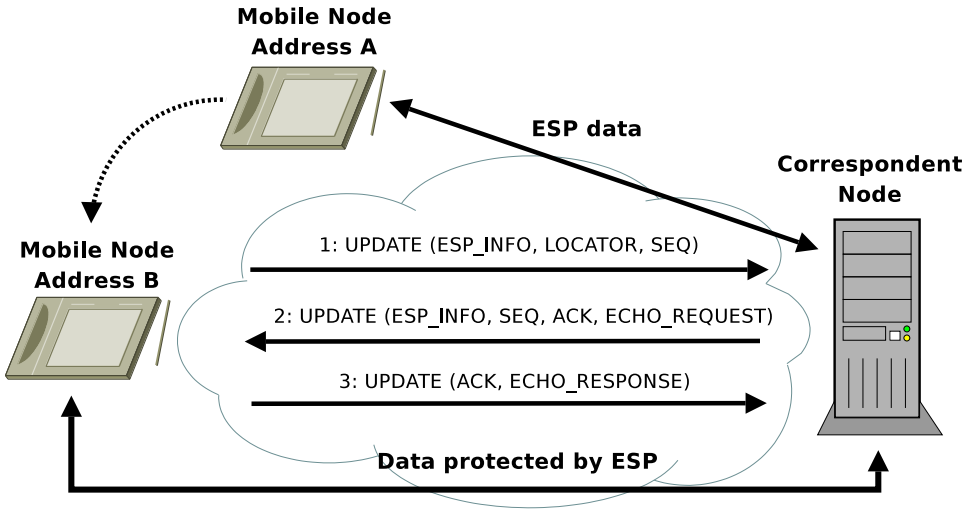


Figure 2.2: HIP mobility update.

ESP_INFO of the peer. Finally, the mobile client is expected to acknowledge the message from its peer and return the content of the echo message. When the peer host gets this response, the new IP address of the client is marked as verified and the update procedure is completed [79].

In some cases a host might have more than one IP addresses associated with a certain interface or even several interfaces attached to different access points. Such host is often referred to as multihomed and is able to maintain multiple connections over distinct paths. HIP provides an opportunity for a host to inform its peers about available interfaces through the use of signalling messages described above. The peer hosts update the appropriate HI↔IP bindings and verify each of the IP addresses of the multihomed host by sending echo requests and waiting for correct replies.

The base specification of HIP mobility and multihoming is presented in the RFC 5206 [79]. Besides that, several extensions have been proposed over past years in research to complement HIP with micromobility [84, 96, 108] and network mobility [41, 85] support. Some of these studies are mentioned under related work in Section 3.2.

2.4 Symbian OS networking architecture

A part of our research is devoted to performance evaluation of the Host Identity Protocol on Symbian OS S60 smartphones. This section gives to the reader brief details about Symbian networking architecture important from the perspective of porting an open source software (OSS) to it.

Symbian OS networking architecture is dominantly based on the client-server communication model. Applications written as clients usually connect to and exchange data with particular servers (socket, telephony, serial communications, etc.). A server then communicates with low-level entities such as logical and physical device drivers (LDD and PDD) via an interface of server plug-in modules. Different module types include CSY (serial communications server), TSY (telephony server), PRT (socket server), and MTM (message type) modules. Clients do not directly access the plug-in modules. The latter are instead loaded by the server on demand [24].

The socket server (ESOCK) is responsible for socket APIs on Symbian and provides two types of interfaces: a BSD-like C socket API (based on Open C plug-in [27]) and an alternative Symbian-native C++ socket API. The socket server works with PRT protocol modules that are supplied in a form of dynamic link libraries (DLLs) with a .PRT extension. The TCPIP.PRT module comprises support of IPv4/v6, ICMP, TCP and UDP, as well as DNS infrastructure [24].

Our HIP implementation for Symbian OS is entirely based on the Open C plug-in that provides support of many standard C socket APIs. The Open C plug-in serves as an interface between the HIP daemon application and the PRT protocol modules in the Symbian networking stack.

3 Related Work

A number of studies evaluated different network-layer security and mobility mechanisms running on traditional computers [38, 48, 107]. Several research projects studied security aspects of communications in sensor networks from the perspective of tiny participating nodes [31, 100]. Other studies were devoted to protection of sensitive health care data being transferred between remote monitoring mobile devices [66]. This section highlights interesting related research performed on IP security and mobility, focusing on mobile lightweight devices.

3.1 Studies on IP security

In the following sections we survey several research studies that aimed at securing IP layer communications of mobile devices by means of elliptic curve, symmetric and asymmetric cryptography. We also give a word to some alternative approaches such as those using hash chains and overview the Internet Key Exchange protocol.

3.1.1 Elliptic Curve Cryptography

Malhotra *et al.* [66] evaluated the use of Elliptic-Curve Cryptography (ECC) to protect sensitive health data in a patient monitoring system on a PDA. The authors propose a secure protocol that aims at data encryption and user authentication. According to theoretical background given by the authors, ECC has a great advantage over RSA security algorithms because it consumes less memory and processing time. As an example, a 1024-bits RSA key's size would be equivalent to the size of a 160-bits ECC key. This, in authors' opinion emphasizes benefits of using ECC on constrained mobile devices [66]. Unfortunately, besides referring to other studies in the ECC field, the authors do not provide empirical comparison results between ECC and public-key cryptography that would have been useful to evaluate different schemes on lightweight devices. However, the article is valuable as it shows some particular performance results of ECC on a PDA SPV M5000 running at 520 MHz with 64 MB of RAM. Interestingly, even with such considerable hardware resources for a mobile device, ECC encryption over WLAN and 3G produced notable latency compared to plain data communications. As the results indicate, the total time to communicate a small text message over the implemented ECC protocol varies from 8 seconds in WLAN to 11 seconds in 3G network. ECC signature verification takes

on the average 6-7 seconds of the total time. ECC security algorithm over WLAN in this example study brings a 16-fold overhead comparing to non-ECC case (8 versus 0.5 seconds) [66].

3.1.2 Symmetric versus public-key cryptography

Continuing from the previous section on Elliptic Curve Cryptography, a work by Wang *et al.* [100] presents a comparison between symmetric cryptography and ECC-based public-key cryptography in sensors networks. The main objective of Wang *et al.* in this study was to address important security aspects of large wireless sensor networks with tiny sensor nodes, such as user authentication and data access control. Namely, the authors aim at achieving protection from several attack types including message eavesdropping, traffic monitoring, sensor compromising and flooding attacks. They design a set of asymmetric and symmetric cryptography schemes for establishing a shared key between communicating entities and compared them on commercially available *MICAz* pairwise sensor motes. These devices include a 8-MHz CPU, 128K of flash memory and 4K of RAM. In the experiments the authors evaluate a pairwise key establishment process and authentication of a mobile user to a sensor mote from the perspectives of processing time, memory overhead, the amount of transferred messages (message complexity) and energy consumption. Based on the performance results obtained in several tests, Wang *et al.* conclude that ECC-based public-key cryptography has more advantages over symmetric cryptography in terms of message complexity, use of the memory and security [100].

A recent study by Haque *et al.* [31] introduces a public-key based mechanism to protect node-to-node communications in wireless sensor networks. In particular, the authors consider a healthcare system scenario where small sensors transmit sensitive data between each other and to mobile terminals, with the help of a secure base station. The presented approach comprises two components: a) a key negotiation scheme to generate a shared secret between a sensor node and the secure base station; b) a decryption key derivation mechanism used by a receiver node for each particular sender. In this scheme, the secure base station serves as a key generation and management entity. It first establishes a pairwise session key with the sender of a message, which is used for message encryption, and then securely transmits a correspondent decryption key for this message to the receiver. Based on simulation, the authors compare their own proposed solution with two other security schemes in terms of the energy consumed for communications, which is an important metric in wireless sensor networks. The results indicate that the proposed architecture

shows better performance than one system, whereas consumes more energy than another (10.1 mJ for sender and 5.7 mJ for receiver). However, the authors do not show any estimates for key handshake (based on public-key cryptography) duration between a sensor node and the base station. Although the system is presented as a very scalable end-to-end security scheme, each sensor-to-sensor communication requires establishing a secure context with the central base station, which produces additional costs.

He and Zhang [32] propose a protocol for asymmetric authentication of end hosts, one of which is a weak mobile client connected over the air to a wired service provider in the Internet. The presented approach is based on delegating part of the computationally expensive cryptographic operations from a mobile client to a third party that is a representative of the mobile node in its home network. The design allows to authenticate the mobile client and the service provider to each other via a home network proxy, whilst not revealing the session key to the latter. While the idea of using a home agent to achieve an extra functionality for mobile clients is not novel, the protocol designed by He and Zhang deserves proper attention since it addresses an essential property such as authentication. On the other hand, a number of issues remain to be unclear, for instance, potential client movements and mobility, and the overhead introduced by a home representative. In addition, the authors mention that running asymmetric cryptography on lightweight devices is expensive without providing any actual figures [32].

3.1.3 LHIP

Lightweight HIP or LHIP [33, 34] originally derived from the early thoughts on computational complexity of public-key cryptography in HIP when used on resource-constrained nodes. The motivation for this work was further reinforced later with availability of the first empirical HIP measurement results on Nokia 770 Internet Tablet [54]. In Lightweight HIP, Heer suggests a lightweight authentication extension for the Host Identity Protocol, which uses hash chains instead of computationally expensive asymmetric cryptography. LHIP achieves up to two orders of magnitude reduction of HIP computational cost at the expense of public key authentication, thereby making the protocol more suitable for mobile appliances with low resource base [33, 34].

3.1.4 IKE and MOBIKE

Internet Key Exchange version 2 (IKEv2) protocol is standardized in RFC 4306 [50]. The purpose of IKE is to authenticate two communication entities to each other by exchanging theirs keys and to create a pair of Security Associations (SAs) to be used then with IPsec (IP security) ESP (Encapsulated Security Payload) and AH (Authentication Header) protocols [51, 52]. In addition to general IKE specifications, there has been a number of efforts on how to tailor it to particular needs of embedded systems (e.g., a lightweight IKE) and mobile environments (e.g., MOBIKE). Lim *et al.* [65] aim at integrating IP security into embedded network components, such as routers. Based on IKEv1 standard, the authors design and implement a lightweight IKE protocol extension that has a minimal set of standard RFC functions but is interoperable with other IKE protocols and thus can be used to establish SAs with different clients. However, since the IKEv1 has been enhanced and obsoleted by the IKEv2, presented lightweight IKE extension needs to be reimplemented to be compatible with the latest standards.

MOBIKE [86] is an extension to IKEv2 that provides mobility and multihoming support so that established IKE SAs can be updated when the IP address of one participating host changes. In the simple scenario this may happen due to terminal movements and change of an access network. In a more sophisticated scenario MOBIKE can support multihoming, i.e., the use of different network interfaces at the same time, as well as IP interfamily (IPv4 and IPv6) handover. MOBIKE does not support any rendezvous service and this prevents simultaneous IP address change on both hosts. This is one important distinction that differentiates MOBIKE from the Host Identity Protocol [73]. Nevertheless, IKE and HIP have much in common. Both aim at establishing a secure communication context between two hosts and generating keying material for subsequent use by IPsec. This generated substantial interest in functionality comparison and evaluation of security level of both protocols. Jian *et al.* propose to replace the HIP base exchange with an IKE extension as a mechanism to eliminate some security risks [44].

3.2 Research on IP mobility

A number of previous research projects have studied mobility management issues in next generation wireless networks. Nevertheless, to our best knowledge, there have been little or no efforts to evaluate performance of different mobility mechanisms running particularly on lightweight mobile devices taking into account their spe-

cific constraints and use cases. Such small appliances belong to a group of devices that, due to their usage patterns, i.e., frequent movements, require strong mobility support. On the other hand, implementing secure mobility might easily stress lightweight hardware. Our preliminary experiments with HIP mobility on Nokia 770 presented in Section 6.3.6 indicate that secure mobility produces certain costs to a mobile handset in terms of computation time, as compared, for instance, with a conventional laptop.

Thus, we believe that evaluating performance of different mobility extensions on lightweight hardware is important as it would provide a good basis for choosing the most appropriate mechanism for a particular type of device and/or application under particular circumstances. In case of hybrid mobility management concepts comprising multilayered techniques, it is important to balance computational load on a mobile client and in general avoid potential overhead, which might derive from cryptographic operations involved with signalling updates, such as in the Host Identity Protocol.

3.2.1 Mobile IP and HIP

In this thesis, besides presenting our practical experience with HIP mobility, we find it necessary to look at the related studies and overview some essential approaches that make host mobility and multihoming possible. Mobile IP (MIP) [88] has been around for a long time and, despite of the bottlenecks (such as, e.g., triangular routing), represents one of the popular approaches to address mobility in the existing Internet architecture initially designed for stationary hosts. Mobile IP protocol is intended to run with minimal changes to the present end hosts in the Internet by introducing a *home agent* and a *foreign agent* for a mobile node. These agents continuously maintain a binding between each other through a tunnel so that the home agent is always aware of the current IP address of the mobile node via its foreign agent. Binding updates are performed either upon changing of the network attachment point by the mobile node or when the binding lifetime has expired. The mobile node is always identified with its home agent and regardless of any physical movements and changes of its network attachment point, the IP address of the mobile node remains the same for respective transport connections. This way IP mobility performed on the network layer is transparent to the upper layers protocols and applications [88].

In research, there can be found several comparative studies and performance evaluations of different mobility protocols on conventional PCs or laptops. Henderson

et al. [40] present experience with the Host Identity Protocol from the perspective of secure mobility and multihoming. The contribution of the paper is significant as it provides a comprehensive comparison of HIP and other mobility and multihoming approaches, as well as reports on a HIP pilot performance and identifies directions for future research. In particular, HIP mobility is compared to the Mobile IPv6 with "route optimization" from the MIPv6 base specification [46]. The authors highlight the main differences in mobility procedures such as the use of MIPv6 Binding Update versus HIP Readdress packets; MIPv6 non-IPsec mode versus HIP tight integration with IPsec; MIPv6 home agent versus the use of directory services in HIP; MIPv6 subnet mobility versus pure HIP host-centered approach; security of MIPv6 versus HIP mobility update mechanisms [40]. It is worth mentioning though that the specifications of both protocols have changed since the publication of this article. MIPv6 route optimization has been updated by "Enhanced Route Optimization for Mobile IPv6" [6]. An enhanced procedure benefits from shortening of the handover latency, raising security and lowering signalling overhead. HIP specifications have matured to RFCs 5201-5207 and some parameters of the mobility mechanism have changed, too. For instance, the *REA* parameter has been substituted by the new *LOCATOR* parameter [79]. Besides this, Novaczki *et al.* [85] have suggested an approach to network mobility based on HIP, titled *HIP-NEMO*, thus extending HIP context from pure host mobility and multihoming.

Jokela *et al.* in an early work [48] compare performance of a vertical hand-off with the HIP and the MIPv6 protocols while switching between a WLAN and a GPRS network. The authors measure a signalling delay in both protocols, starting from the first update packet and ending with the first recovered packet of a TCP data stream. In this study, HIP notably outperforms MIPv6 in terms of the average duration of a mobility update (2.46 versus 8.05 seconds respectively). However, the authors suggest that in theory both protocols should perform similarly and explain their empirical results by a bug in the measured MIPv6 implementation that caused simultaneous use of two interfaces for the same TCP stream (packets and their ACKs) for a certain time span during the update procedure. This in turn resulted in overloading of the GPRS link and dropping and retransmitting some of the MIPv6 signalling packets. In the future, the authors plan to measure other MIPv6 implementations, as well as the enhancements to the current mobility update procedures. The work by Jokela *et al.*, however, does not concern the perspective of the underlying hardware and the types of the communicating hosts we are interested in. It compares the work of two separate approaches to mobility in practice rather than evaluates performance of a single protocol on different hardware platforms.

Nevertheless, the article provides a useful reference material for our research.

3.2.2 Multilayered mobility management

Numerous studies took a multilayered approach to mobility and evaluated feasibility and efficiency of integrating multiple protocols operating on different OSI layers [9, 28, 64, 49, 95, 101]. Of these, considerable research has been conducted on combining network layer mobility mechanisms with mobility on the application layer. The intention of such integration approaches is usually to provide "all-in-one" mobility support for versatile types of applications, improve QoS and shorten time needed to perform a handover. For instance, the authors of [64, 49, 101] proposed a multilayered mobility management approach based on a combination of the Session Initiation Protocol (SIP) [90] and the Mobile IP.

Schulzrinne and Wedlund [92] describe how SIP signalling provides terminal, session, personal, and service mobility for different applications. By drawing a parallel between SIP and Mobile IP and comparing flexibility provided to users, the authors identify the most favourable scenarios for each protocol and conclude that application-layer mobility can successfully be a partial substitution or an extension for network-layer mobility. Despite its well known limitations, MIP is found by the authors to be more efficient for terminal mobility and TCP connections [92].

Similarly with Mobile IP, HIP has become a candidate protocol to combine it with SIP for cooperative mobility management on two layers. Henderson [39] discusses possible integration of SIP and HIP and, among the main benefits, which the latter protocol might potentially bring to the former, lists resistance to DoS attacks, the ability to preserve TCP sessions upon an IP address change, better micromobility management, NAT traversal, and the possibility to integrate some HIP and SIP components (such as a HIP rendezvous server and a SIP proxy). However, the author emphasizes that it is not clear whether all mentioned HIP features, due to their immaturity as of writing the article in 2004, can turn to be advantageous to SIP in the future [39]. Since the time of the Henderson's discussion both HIP and SIP have matured and now open up new perspectives for combining them. HIP, for example, has been specified in the seven RFCs (5201-5207), including the specifications for registration, rendezvous, mobility and multihoming, DNS and NAT traversal extensions. In addition, several proposals have been suggested for HIP micromobility [84, 96, 108] and network mobility [85, 41] support.

Since Henderson, HIP and SIP integration has been further reflected in research. A hybrid scheme called *SHIP* has been suggested by So *et al.* [95]. Based on experi-

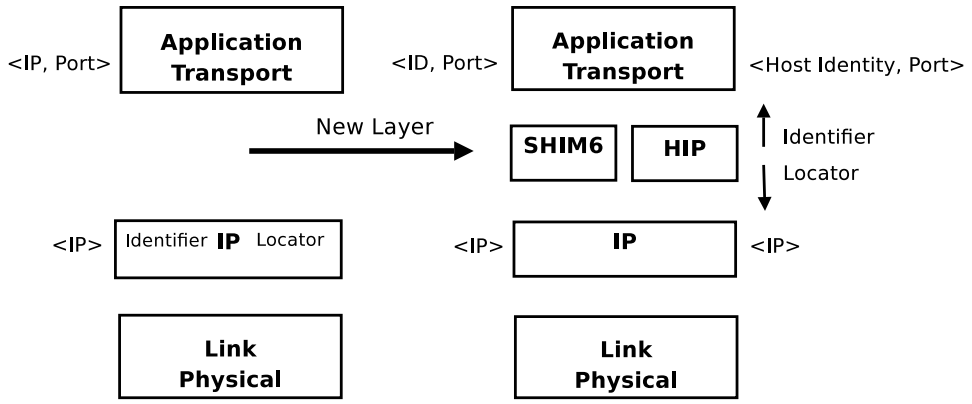


Figure 3.1: SHIM6 and HIP layers in the protocol stack.

mental evaluation, the authors show that SHIP outperforms a hybrid MIP and SIP scheme in terms of signalling overhead and efficiency [95]. A more recent work by Camarillo *et al.* introduces a framework for combining SIP and HIP and emphasizes its advantages [9].

3.2.3 SHIM6

SHIM6 presents an approach that has much in common with the Host Identity Protocol. The core specification of SHIM6 is presented in the Internet-Draft "Shim6: Level 3 Multihoming Shim Protocol for IPv6" [83]. The protocol aims at providing multihoming functionality for IPv6 protocol enabling also reachability and failure detection mechanisms. With multihoming capability Internet hosts are able to share their communication loads between different network interfaces. Besides that, when some of the currently used locators change or stop working, SHIM6 reacts appropriately by switching to new locators seamlessly for upper layer applications and protocols [83].

HIP also provides support for host multihoming using update mechanisms similar to mobility. HIP is based on the concept of locator-identifier split and, similarly with SHIM6, introduces a new protocol layer between the network and transport layers (see Figure 3.1).

3.3 HIP performance evaluation

In addition to theoretical comparison of mobility approaches, Henderson *et al.* [40] present their performance results of HIP running on Dell Latitude laptops. These results are particularly interesting to us in respect to our own empirical study (see Chapter 6) because the CPU clock frequency of the platforms in both studies are in the same range (the Dell laptop had a 266-MHz Pentium II CPU [40] whereas the Nokia 770 is powered by a 220-MHz ARM processor [54]). Obviously, there are other potentially influential factors that, in fact, have been different in the experiments (e.g., Linux kernel version, network connection and the amount of RAM). Nevertheless, the results by Henderson *et al.* are comparable to ours in terms of the total duration of a HIP base exchange for a lightweight initiator and the impact of the puzzle difficulty on the average processing time. While a HIP handshake on the Dell laptop takes 0.95 seconds, the Nokia 770 performs a BEX in 1.40 seconds on the average. Both studies show an exponential growth of the puzzle processing time, which increases dramatically with the puzzle difficulty set over 15 bits. A three-way HIP mobility update takes on the average on the Dell laptop 180 ms and on the Nokia tablet 287 ms. While comparing the total duration of a HIP handshake or a HIP mobility update it is important to know what the Round-Trip Time (RTT) in both experiments accounts for. Unfortunately, Henderson *et al.* do not provide such information, only indicating that the communicating Dell laptops were connected over 10Base-T Ethernet. In turn, in our experiments an average RTT was equal to 2.8 ms, accounting for a negligible part of the handshake duration. Both studies indicate that a large part of the BEX and update time is spent on the cryptographic operations such as signing and verification procedures [40, 54].

Another study by Nikander *et al.* [82] proposes a HIP-based cumulative way to address security, mobility and multihoming in the current Internet. Besides an extensive description of the architecture and its components, the authors present early implementation status and initial results of performing a HIP four-way exchange on two 800-MHz Pentium III machines running NetBSD 1.6 and connected via a 100-Mbps Ethernet. Interestingly, the processing time that includes solving a cryptographic puzzle greatly varies in the experiments even when puzzle difficulty K is under 10 bits (from 300 to 2300 ms). The authors explain such deviation by an indeterministic character of puzzle solving operation, which requires a larger number of measurement repetitions to get more actual and precise results. Varying time needed to solve a puzzle subsequently makes an impact on total duration of a HIP association establishment, which ranges from 600 ms to 3 seconds with K equal to zero and 10 respectively. Further increasing K raises processing time by

a large factor, approaching 100 seconds when K amounts 15 [82]. This exponential growth goes very well with theoretical expectations, the results presented by Henderson *et al.* [40] and our own results illustrated in Section 6.3.2. However, in our measurements on a Nokia 770, an embedded Linux PDA, a boundary, which determines a rapid increase of the puzzle processing time, is K = 15. Even with this value of the puzzle difficulty, the Nokia 770 spends only 1.5 seconds on the average on the respective base exchange phase, which is highly contrasting with the above value of 100 seconds observed by Nikander *et al.* [82]. One potential reason for different results might be differences in the HIP implementations, operating systems and underlying hardware platforms.

Pääkkönen *et al.* in a recent study [87] present detailed measurement results of the HIP-based handovers performed between different access networks and IP address families. In addition, the authors consider various triggers to deliver event information between the layers of the protocol stack, such as changes in IP addresses, routing tables, and on the link layer. In the evaluation, triggering is thus integrated with mobility in a testbed that includes a mobile node, a mobile router, a mobile phone and a correspondent node that are all connected via versatile accesses (LAN, WLAN, 3G and Bluetooth). The underlying test platform is FreeBSD that runs HIP (here HIP4BSD software). The contribution of the study is an extensive evaluation of the impact made by individual components, which constitute to the overall triggering and handoff latency performed with varying access networks and IP protocol versions. For instance, the delivery time of a mobility trigger from the IP to the HIP layer ranges from 49 to 226 ms. HIP handover delays vary from 0.5 to 2.9 seconds for the LAN→WLAN switching and from 1.5 to 2.5 seconds for the LAN→3G handoff. According to the authors' observation, Duplicate Address Detection and Router Discovery increase the handoff latency in case of IPv6 autoconfiguration. Interestingly, processing of a HIP ACK message involving updates to SAs (Security Associations) and SP (Security Policies) on the correspondent node might occupy up to 55% of the total handover duration. With 3G, the major impact is made by the link latency [87].

3.4 Security and mobility issues in wireless networks

In this section we describe selected works in the field of security and mobility in wireless networks that are relevant to our authentication architecture presented in details in Chapter 8. Our original work [61] has much in common with research problems described in this thesis in a way that it addresses security and mobility

issues in wireless LANs by exploiting many of the HIP properties. However, cryptographic operations involved with HIP might negatively affect performance of the whole architecture if it consists primarily of the lightweight components. Not all models of the Wi-Fi access routers, for instance, are equipped with sufficient hardware resources to be part of such an architecture. Section 8.4 focuses on this issue and provides interesting details. Below we refer to several related research areas.

An architecture for secure and mobile Wi-Fi sharing is proposed by Heer *et al.* [35, 37]. *PISA* (P2P Wi-Fi Internet Sharing Architecture) eliminates well-known security risks and attacks in open wireless networks. It also provides a solution for client authentication and mobility, which is similar to our proposal described in Chapter 8. On the other hand, PISA focuses on serving a slightly different role in the Internet, namely implementing secure access control for global Wi-Fi sharing communities. Similar work [23] considers distributed authentication, authorization and accounting (AAA) in community Wi-Fi networks. It concentrates on building trust chains between communicating entities using certificates and certificate authorities (CA). Later valid certificates serve for authenticating the clients, as well as for identifying and validating the decentralized AAA servers.

The concept of a distributed firewall used in our approach is not novel but existed for a long time. Studies [43, 98] discuss the advantages of a distributed firewall over a conventional centralized firewall in changing network topologies. The papers describe key points in implementing a distributed firewall, including a mechanism to enforce network security policy through a policy language. Additionally, a distribution scheme and an authentication technique for network entities participating in the policy enforcement process are presented.

Source address validation is another topic related to our work. Source address validation architecture (SAVA) [103, 104, 105] addresses the problem of source address spoofing on different levels of granularity, from a local subnetwork to autonomous systems.

4 Research Problem

In this chapter we articulate the research problem that motivated us for this work. We discuss three different perspectives of the problem: security, mobility and energy consumption.

4.1 Security perspective

Although there have been several activities on adapting IP technology [19, 17] and some related applications to embedded systems with severely constrained resources, the security aspect of using the TCP/IP stack on lightweight devices is not sufficiently explored. Among other researchers, Abeillé *et al.* [5] note that further studies are necessary to evaluate security mechanisms in the lightweight IPv6 stack implementations dedicated to small objects. CPU, RAM and battery constraints of lightweight mobile devices raise the concern whether IP security mechanisms might be employed there without major modifications. In particular, asymmetric cryptography algorithms implemented in software used to secure communications between mobile resource-constrained devices can easily stress their CPU and memory resources, as well as negatively affect battery lifetime, TCP throughput and Round-Trip Time.

We address this problem by measuring and evaluating performance of different IP security components in the Host Identity Protocol. We analyse the impact made by the heavyweight cryptographic operations on the constrained resources of mobile devices, such as CPU load, RAM usage, and battery lifetime. We also assess the impact made by the limited processor power and memory constraints on the duration of certain protocol parts including HIP base exchange, HIP mobility update, puzzle solving procedure, and generation of a public-private key pair. In addition, we evaluate the effect of IPsec ESP encryption on packet latency and TCP throughput. This allows us to make recommendations on suitability of unmodified HIP for lightweight class of mobile clients.

4.2 Mobility perspective

Mobility of a device denotes its freedom from a fixed network attachment point. Thus, efficient and reliable mobility solutions are necessary for this class of devices. If mobility does not work properly, the device loses its notion of being *mobile*. You

can still carry it wherever you go, however, most likely, you will not be able to utilize its full potential and keep your Internet sessions ongoing due to the limited network coverage or changes of the network attachment point. On the other hand, for a mobile client connected to the Internet wirelessly, i.e. via a cellular network or a WLAN, the threat of attacks and the probability to be compromised are raising compared to the wired networks. Hence, mobility needs to be secure.

Numerous solutions proposed in the literature range from network and application-layer mobility to multilayered approaches [40, 82, 48, 49, 64, 95, 39]. However, little knowledge is available on how well these solutions scale to the nature of the embedded platforms that have distinct requirements from the desktop computers and often behave differently. A mobile phone changes its network attachment point more frequently than, e.g., a laptop or a PC. In addition, limited resources of an embedded platform underlying a mobile phone may produce a need to adjust or tailor existing solutions to achieve better performance.

Our objective with respect to mobility in this work is to identify potential issues of running the IP-based secure mobility and multihoming protocols on lightweight clients with constrained resources. Based on our experiments, we show that a HIP mobility update lasts on a Nokia 770 Internet Tablet almost three times longer on the average than on a laptop with similar RTT for both clients. Long duration of a network-layer update procedure on small mobile devices might produce implications for upper-layer applications. For some applications, there are strict timeouts and requirements for the duration of an update procedure to be able to sustain an IP address change. This motivates us to look at performance issues of secure IP mobility.

4.3 Energy perspective

Power consumption is another critical issue for a mobile, battery-powered device. A concept of accumulating power from a small battery rather than from a fixed power outlet allows devices to be wearable, which partly constitutes the notion of *mobility*. The challenges, however, arise from limited battery capacity that often prevents the applications, especially involving wireless data transmission, from being active for a long time. If an appliance needs to be recharged often, one can hardly call it *mobile* as it requires frequent periodical wired connection to an electricity supply. Coping with battery constraints on lightweight mobile handhelds is not an easy task from the engineering perspective. Constantly increasing battery capacity cannot be a long-term solution to the problem as it produces more heating to a mobile device

making it less comfortable to use.

The use of wireless connection by a mobile device and computationally expensive cryptography operations increase energy consumption and shorten battery depletion time. One of our objectives in this thesis is to evaluate the impact made by the IP security and mobility mechanisms on power consumption and battery lifetime of a lightweight mobile device. In our experimentations, we measure the power consumption of the Host Identity Protocol, which provides secure mobility by means of public-key cryptography and IPsec ESP encryption.

5 Methodology

In this chapter we present the methodology for our work. First, we state the main research methods. Next, we describe the primary research tools used in our experimentations. Finally, we define the core assumptions in our work and limitations of the research prototypes.

5.1 Research methods

The primary research methods used in this work have been literature study and measurements performed in a real test network. Through the literature study we obtained the insights into the research problem, evaluated the contribution of the related work in the same field, and gained necessary background knowledge. The survey of the related studies allowed us to draw some parallels with our own empirical performance results. The empirical results were conducted in a number of experimentations involving several mobile clients communicating with other hosts and between each other in a variety of scenarios. We measured a set of different parameters related to the Host Identity Protocol and two general network characteristics. To get statistically more accurate results we repeated each measurement a number of times. We then analysed the results from different perspectives and made conclusions about feasibility of using unmodified HIP and IPsec on lightweight mobile clients.

5.2 Research tools

This section presents a set of the tools that we have been using to accomplish empirical part of our research. We start with a description of the development environment that allowed us to port HIP to Maemo and Symbian mobile platforms. We then introduce our test network and its components followed by a list of the measurement tools.

5.2.1 Development environment

In the development phase, we ported an existing HIP software to two types of mobile platforms, *Linux Maemo* and *Symbian S60*. For Maemo, we took the existing HIPL (HIP for Linux) protocol implementation and compiled it for the Nokia 770

Internet Tablet using the Scratchbox cross-compilation toolkit and a Maemo SDK. Further details on this process can be found in Section 6.1. For Symbian, the porting procedure has been far more difficult due to the restrictions of public Symbian SDKs and the platform-dependent HIPL code. Before we were able to run the protocol and measure its actual performance on a Symbian smartphone we needed to go through a number of steps that are described in details in Section 7.1. As the development environment we used an S60 3rd Edition Platform SDK for Symbian OS, the Carbide.c++ IDE and the Open C SDK plug-in for S60 3rd Edition SDK. An important role in the whole porting process to Symbian was given to the Symbian S60 Emulator supplied with the SDK. With the emulator, we were able to debug runtime code errors with less effort and test the main protocol operations before measuring their performance on the actual mobile phones, Nokia E51 and N80.

5.2.2 Experiment setup

Measurements have been an essential part of our experiments to conduct empirical results about performance of public-key operations and IPsec ESP encryption on different mobile platforms. For the measurements we constructed a test network consisting of a wireless access router IEEE 802.11 b/g connected with a server via a network switch. Mobile clients such as a Nokia 770 Internet Tablet, an IBM laptop and a Symbian-based Nokia E51 smartphone were connected to the network via their wireless interfaces. The general network view is presented in Figure 5.1.

5.2.3 Measurement tools

To measure different protocol and network performance metrics we used a number of tools and utilities, including *iperf, ping, tcpdump, ping6* and *Wireshark (Ethereal).* With *iperf* we measured the TCP throughput, with *ping* and *ping6* we measured the RTT and triggered the HIP base exchange. We used *tcpdump* and *Wireshark* network analyser to capture traffic, analyse individual data packets and record their arrival and departure times to calculate the duration of particular operations. In addition, we created a set of shell scripts to automate measurement process and ease repetition of the measurements. To analyse hardware resource utilization during the protocol operations we used *Nokia Energy Profiler* with the Nokia E51 phone. With the Nokia 770, we originally used an external multimeter to measure current consumption of different applications running on the Internet tablet by connecting its battery pins with the multimeter probes. With the OpenHIP implementation, we used time stamps in the code to measure the delays of certain protocol operations.

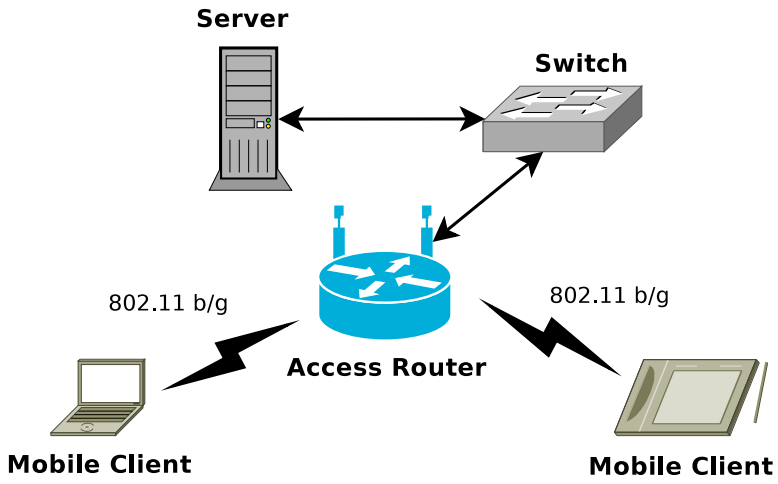


Figure 5.1: General view of the network setup.

5.3 Limitations and assumptions

When measuring duration of a HIP mobility update we "artificially" triggered mobility by removing the current IP address of the Nokia 770 and adding a new IP address manually. In this way we emulated a horizontal handover. Our assumption was that regardless of the way, in which the IP address changes, HIP notices such a change and initiates a three-way HIP mobility handshake. By the handshake duration we mean the time needed for a mobile device to exchange three mobility update packets with its peer before the data transmission continues.

As will be in details illustrated in Section 7.1.5, our Symbian HIP ports have several limitations. First, we implement only the base operations of HIP including the HIP base exchange and the security association establishment. Second, data encryption by the IPsec ESP protocol is not realized due to the limitations of the public Symbian SDKs and the complexity of implementing a substitute to the Linux TUN/TAP device driver on Symbian. Finally, our Symbian ports can be run only on a limited number of the end user smartphones. Symbian platform security concept necessitates signing of each application package against a specific phone's IMEI before the package can be installed on the device. This makes debugging on the target hardware inconvenient, requiring numerous rebuilding and resigning steps, and, at the same time, complicates the large-scale deployment of the software.

6 Performance of Host Identity Protocol on Nokia Internet Tablets

In this chapter, we report our experience with running the Host Identity Protocol on a Linux-based Nokia Internet Tablet. The chapter is based on our article *Performance of Host Identity Protocol on Lightweight Hardware*[7] [54]. Section 6.1 briefly describes the Nokia 770 Internet Tablet hardware and software, as well as our port of the HIPL implementation. In Section 6.2 we present the components of our experimental testbed. Section 6.3 contains measurement results of HIP over WLAN with a Nokia 770 tablet in a set of scenarios. In particular, we measure the duration of a HIP base exchange, a HIP mobility update, the data throughput and the latency of a wireless network, as well as the impact of the protocol operations on power consumption of a Nokia 770. We analyse each type of the measurements and conclude about potential HIP implications for similar mobile devices with restricted resources. Finally, Section 6.4 summarizes our performance evaluation on Nokia 770 with a list of recommendations.

Our choice of Nokia 770 Internet Tablet as a target device for experimentation had been supported by several factors. First of all, it is a resource-constrained PDA that provides a good example of lightweight hardware for assessing performance of IP security and mobility. Second of all, such a handheld ideally represents a mobile client constantly moving across the Internet and changing its network attachment point. In this approach, the tablet would be a desired target to test a mobility protocol, e.g., the mobility extensions of HIPL. Next, at the time of the experimentation, Nokia 770 was gaining its popularity among both end-users and developers, which resulted in a number of multimedia applications that might potentially utilize the benefits of HIP. Finally, the embedded Linux OS running on Nokia 770 made it easier to port the existing HIPL software from desktop to mobile platform.

6.1 HIP on the Nokia Internet Tablet

This section outlines the Nokia 770 technical specifications, as well as describes the porting process of the HIPL implementation.

The Nokia 770 Internet Tablet is a Linux-based handheld with a high-resolution touch screen display, built-in WLAN and Bluetooth support. Mainly designed for

[7] © 2007 ACM. Used with permission.

easy Web browsing, the tablet is also convenient for Internet telephony and instant messaging, reading emails and documents, and delivering media content. In its core, Nokia 770 has a Texas Instruments (TI) OMAP 1710 CPU running at 220 MHz. The device comes with a 64 MB DDR RAM and is powered by a 1500-mAh Li-Polymer battery. The operating system is a modified version of Debian/GNU Linux. For our experiments, we used a release version known as the Internet Tablet OS 2006 edition. It has a GNOME-based graphical user interface and runs a 2.6.16 series Linux kernel.

Porting HIP to the Nokia 770 Internet Tablet consisted of two main stages, configuring and compiling the Linux kernel, and building the protocol software for the Nokia 770. Since the handheld is running an embedded Linux, we used an existing Linux implementation of the protocol, HIP for Linux (HIPL), developed at Helsinki Institute for Information Technology. Although the HIP daemon and other utility programs of HIPL are the userspace applications, several modifications to the Linux kernel were necessary to support HIP at the time of the experiment. In particular, we had to apply an IPsec BEET patch, configure support of the IPv6, IPsec, AES, 3DES, and SHA1 algorithms and recompile the kernel for the ARM platform. To build the HIPL userspace applications and the Nokia 770 Linux kernel we used a cross-compilation environment Scratchbox that emulates the ARM environment on a PC and allows compiling the applications, which later can be installed on a real device.

6.2 Test environment

We performed our measurements on a Nokia 770 Internet Tablet (from now on - the Tablet) and an Intel Pentium 4 CPU 3.00 GHz machine with 1 GB of RAM (the PC) connected to each other via a switch and a WLAN AR in our test network (see Figure 6.1). The network provided both IPv4 and IPv6 addresses. The wireless AR supported IEEE 802.11g standard and WPA (Wi-Fi Protected Access) encryption. All communicating parties used the same implementation of HIPL. To better indicate the Tablet's performance level we repeated our measurement scenarios with a more powerful, 1.6 GHz IBM laptop (the Laptop) connected to the PC over the same wireless link as the Tablet. Through comparison we evaluated the impact of the Tablet's lightweight hardware on the maximum achievable data throughput, latency, duration of the base exchange and mobility update operations.

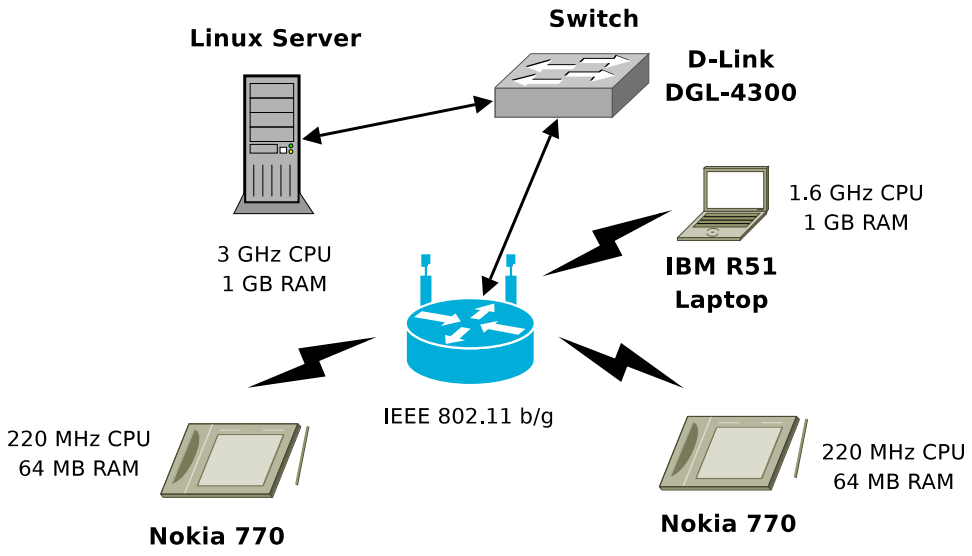


Figure 6.1: Test network with Nokia 770.

6.3 Experiment results on Nokia 770

This section presents the results of our experiments with the Host Identity Protocol on the Nokia 770 Internet Tablet. First, we introduce the platforms and the network environment we used. Then, in the following subsections we report the measurement results and their interpretation.

6.3.1 Duration of a HIP base exchange

A HIP association is set up by exchanging four control packets between communicating hosts. The purpose of measuring the HIP base exchange time was to determine the duration of various BEX stages such as generating and processing the HIP control messages by the Tablet in comparison with the Laptop. The measurement was performed using a script that established a HIP association 50 times in a number of scenarios, which were distinctive from each other by the participating mobile device (Tablet or Laptop), by the IP address family (IPv4 or IPv6) and by the algorithm used (RSA or DSA). Since we did not find significant differences between IPv4 and IPv6 performance we present only the results with the RSA HITs mapped to the IPv6 addresses of the hosts.

Figure 6.2 depicts the times that were measured in our experiments. We leave

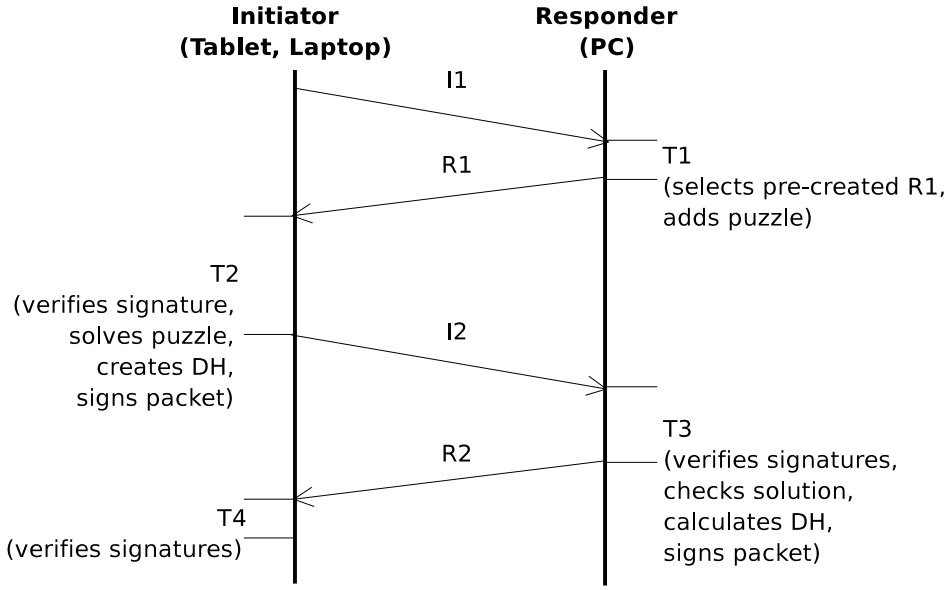


Figure 6.2: Time spans measured on the Initiator and the Responder.

out I1 packet generation time due to its insignificance (the parameters of the I1 packet only include Initiator's and Responder's HITs, which are not signed). T1 represents the time for the Responder to process an I1 packet and generate an R1. According to the HIPL implementation, Responder does not spend much time for this phase since it chooses pre-created and signed R1 messages and adds a puzzle to them just before sending the packet to a network. The next time, T2, contains a number of CPU-intensive cryptographic operations such as generating and verifying signatures, calculating a Diffie-Hellman (DH) session key. During this stage the Initiator must also solve the challenge it received from the Responder. T3 indicates the time needed by the Responder to process an I2 packet that involves the puzzle solution check, Initiator's public key verification and computation of the DH session key. If the puzzle was solved correctly, Responder generates an R2 message and signs it. Finally, during T4 the Initiator processes the R2 packet and completes the BEX. At this point, the HIP association is established.

Figure 6.3 illustrates T1, T2, T3 and T4 times as well as the total duration of the HIP base exchange. We compare the results for two different HIP associations where the Initiators are Tablet and Laptop with the PC acting as the Responder. Thus, T1 and T3 times are measured for the PC whereas T2 and T4 times correspond to both Tablet and Laptop. As the figure indicates, the Laptop greatly outperforms

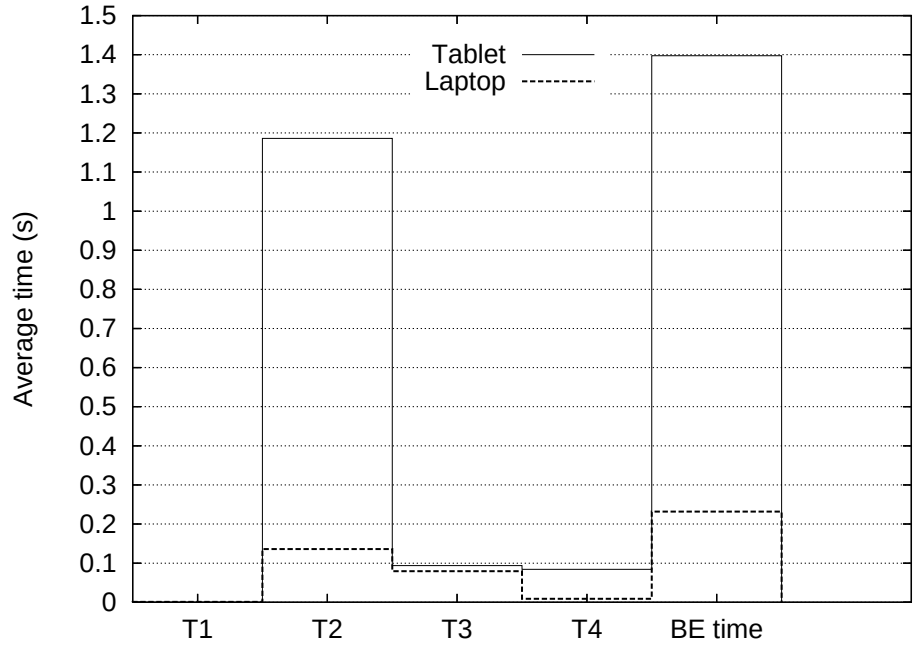


Figure 6.3: Duration of HIP base exchange stages for Tablet and Laptop.

the Tablet for all operations involved with BEX. T2 time for the Tablet is nearly 1.2 seconds which is significantly longer than the respective one of the Laptop (0.14 seconds). The majority of T2 is spent by the Tablet on the operations with the public key signatures and generation of the Diffie-Hellman session key. This processing time heavily depends on the length of a public key and the DH Group ID. For our base tests on Tablet and Laptop we used the RSA key size of 1024 bits and 1536-bit DH Group.

The next test established a HIP association initiated by the PC while the Tablet acted as the Responder. The results suggest that the base exchange time is independent of whether the PC or the Tablet initiates the handshake. In both cases, having the precreated R1 packets, the base exchange lasts around 1.4 seconds.

Although 1.4 seconds to perform a HIP handshake between the Tablet and the correspondent PC might be acceptable for users and applications, HIP communication of two lightweight devices produces a higher delay. The BEX duration for a Tablet-to-Tablet scenario is over 2.6 seconds. The Tablet spends a similar period of time in T2 and T3 phases. The amount of work by the Tablet-Initiator during the phase T2 is analogous to that performed by the Tablet-Responder during the phase T3. The

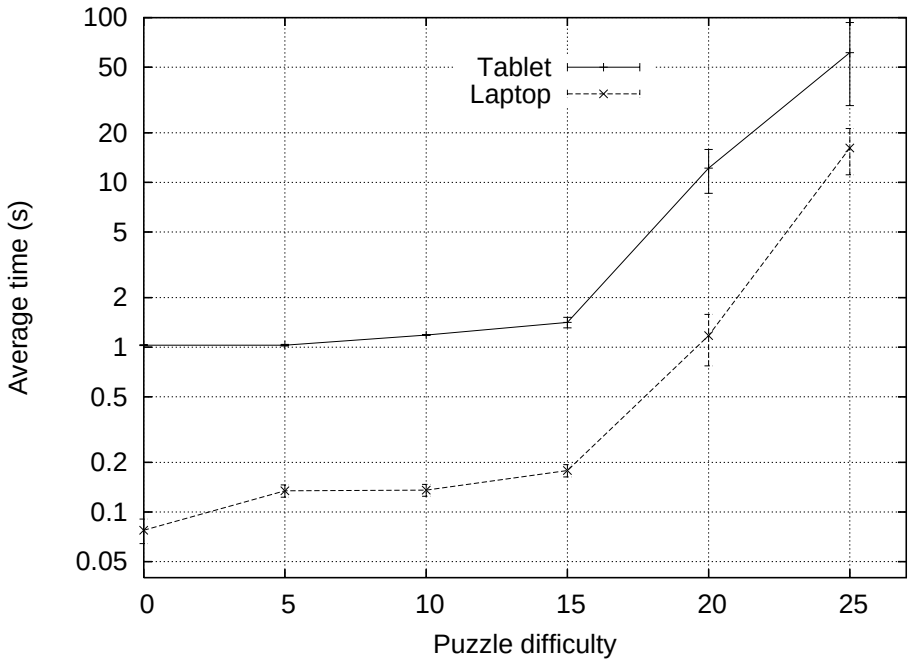


Figure 6.4: T2 processing time versus puzzle difficulty.

only difference is that in T2 the Initiator spends the time for solving a cryptographic challenge whereas in T3 the Responder is supposed to verify the solution to that challenge and also validate the Initiator's HMAC signature. Otherwise, the same operations on public key signatures and Diffie-Hellman keys are carried out by both parties. Later, in the next section we will show that solving a puzzle with difficulty of ten makes a minimal impact on the T2 processing time. Considering this fact and also that puzzle solution check and HMAC validation in T3 are not computationally expensive we believe that the major influence on the BEX parts and the total BE time is exerted by cryptographic operations costly for Tablet's CPU. Such operations include signatures verification and generation, as well as computation of the Diffie-Hellman session key.

6.3.2 Puzzle difficulty

Upon receiving an R1 packet, the Initiator is expected to solve a cookie challenge (puzzle) it gets from the Responder. This is done to protect the Responder against possible Denial-of-Service attacks by compelling the Initiator to spend a certain

Table 6.1: Median and average T2 with standard deviations for varying puzzle difficulty.

	T2 Median/Average±Stdev (sec)			
K (bits) →	5	10	15	20
Tablet	1.03/1.03±0.03	1.19/1.19±0.02	1.33/1.41±0.28	9.06/12.21±9.72
Laptop	0.13/0.14±0.03	0.13/0.14±0.03	0.16/0.18±0.04	0.83/1.20±1.01

amount of CPU cycles to find a right answer. Depending on the conditions, i.e., on the trust level between the communicating endpoints, the Responder has an opportunity to adjust the puzzle difficulty to be solved by the Initiator [72]. The difficulty (K) is represented by a number of bits that must match in a hash output sent back to the Responder. In the presented scenarios the default puzzle difficulty of ten was used. To see how the duration of the base exchange is affected by the puzzle difficulty we measured the time T2 with varying value of K.

Figure 6.4 illustrates this dependency for the Tablet and the Laptop and shows that the time needed to solve the puzzle grows exponentially with increasing its difficulty. The graph depicts the average T2 processing time for the puzzle difficulty ranging from 0 to 25. The number of runs in each experiment was 30. In Table 6.1 we present the mean values and the standard deviation of the T2 processing time that includes the time needed to solve a puzzle by the Tablet and the Laptop. The tabulated results show a substantial increase of the standard deviation with the growing puzzle difficulty. As was noted by Nikander *et al.* [82] in their study of HIP, the reason likely comes from the indeterministic character of the puzzle solving procedure, measuring which requires a larger amount of runs.

An interesting point that we observed in our experiment is that the processing time starts rising dramatically when the puzzle difficulty is set to 15. Prior to this value the effect of increasing the difficulty level is tiny. There is a little difference between the processing times measured for the K values of zero and ten as compared to the T2 value itself of approximately 1 second. This consequently means a minor influence of the puzzle solving time to the total BEX duration in our measurements with the puzzle difficulty of ten.

There is a time limit during which the Initiator must find a solution to the challenge. With Nokia 770, setting a high value of K by the Responder would not be possible

since the Tablet's CPU will spend a long time to solve such puzzle. For example, a puzzle difficulty of 20 would keep the Tablet's CPU busy for over 10 seconds which is unacceptable for most applications. The Laptop, in contrast, would solve a similar challenge in 1.3 seconds on the average. Balancing between the puzzle difficulty and the time limit during which a correct solution is valid for the Responder might be an issue when using the lightweight hardware in a hostile environment with a low level of trust.

6.3.3 Diffie-Hellman

The Diffie-Hellman (DH) key exchange protocol is used in HIP to exchange the public keys of the hosts and produce a session key for the Initiator and the Responder. A piece of keying material is then generated from the session key and is used to create the corresponding HIP associations by the communicating parties [72]. The Responder includes in the R1 packet one or two its public DH keys. Upon receiving the R1 message with two DH values, the Initiator is supposed to select one that corresponds to the strongest DH Group ID it supports. Using different DH Groups makes it possible to affect the generation time of the DH session key and as a result the total duration of the HIP base exchange. In reality, this means an opportunity for a server to offer smaller DH public values to lightweight clients that are not powerful enough or if the security is not of critical importance.

We measured the T2 processing time containing the generation of the DH session key by the Initiator-Tablet and the Initiator-Laptop. The average T2 times for the different DH groups are plotted in Figure 6.5. The graph shows an exponential growth in the processing time as the DH group ID increases. When using the weakest 384-bit DH Group, the Tablet is able to complete the T2 phase in less than 130 ms on the average. This reduces the four-way base exchange to some 200-300 ms with the PC as the Responder. With the 768-bit DH Group, T2 processing time for the Tablet is slightly higher and amounts to 234 ms, resulting in 340 ms on the average for the total duration of the HIP BEX with the PC. However, switching to the 1536-bit DH Group for better security, produces a longer delay close to 1 second on the average. Further increasing the DH modulus length to 3072 and 6144 bits (which might be required under attacks) is not feasible for the Tablet as it results in the tremendous delays for the applications (over 5 and 35 seconds correspondingly). In comparison with the Tablet, the Laptop is capable of handling the stronger encryption and spends less than 0.66 seconds on the average to compute the session key with the 3072-bit DH Group.

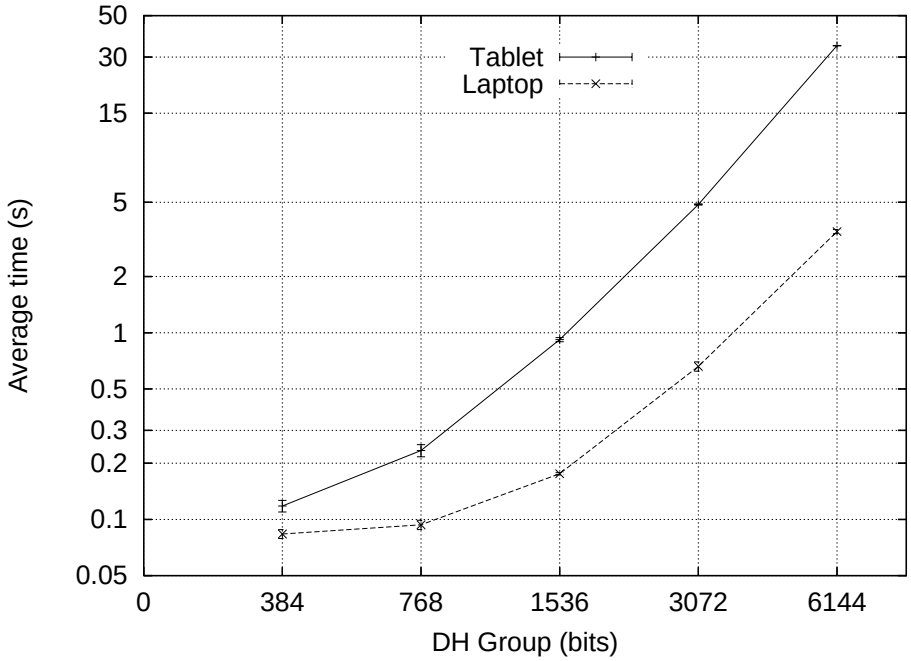


Figure 6.5: T2 processing time with different DH groups.

Our Diffie-Hellman measurements were conducted with a HIPL code snapshot as of May 2007 running on the latest version of the operating system on the Tablet. The DH experiment was also performed in a test network different from the one used for the rest of our HIP measurements. We see these factors as a reason for a difference of the results in the processing time of the HIP control packets on the Tablet (see, for example, Figure 6.4 and Figure 6.5).

6.3.4 Round Trip Time

The RTT (Round Trip Time) equals the time for a packet to travel from a node across a network to another node and back. Our tests evaluate the effect of HIP and, in particular the IPsec BEET mode, on RTT. The tests used the *ping6* tool for sending the ICMP messages over HIP (messages encapsulated with ESP) and over plain IP. We measured RTT in several scenarios including the Tablet, the Laptop and the PC acting as the HIP hosts. The number of runs in each test was 100. Table 6.2 contains the median values of RTT measured over plain IPv6 and over HIP. We calculated the median values instead of the mean values and the standard

Table 6.2: Median and average RTT with standard deviations for Tablet and Laptop.

RTT	**Median/Average±Stdev (ms)**		
	IPv6 (64B)	IPv6 (116B)	IPv6/HIP
PC→Tablet	2.08/2.22±0.47	2.25/2.36±0.42	2.75/2.94±0.93
Tablet→PC	1.80/1.90±0.33	1.80/1.90±1.24	2.50/2.75±1.35
PC→Laptop	0.95/1.03±0.34	0.99/1.05±0.31	1.08/1.18±0.24
Laptop→PC	0.96/1.07±0.34	0.97/1.07±0.43	1.08/1.21±0.50

deviations because the RTT distributions in our experiments had several outliers. The first outlier in each test was the first RTT value, which was large due to a HIP base exchange and an ARP query performed upon the first connection. This value was excluded from the distributions since our intention in this experiment was to assess the impact of the IPsec encryption on the RTT. However, the RTT distributions in some of the tests had other outliers not connected to the connection establishment. For instance, the tests IPv6(116B) and the IPv6/HIP in the Tablet-to-PC scenario contained one additional outlier caused by an unidentified reason. To identify the exact reason for this outlier we would have had to repeat our tests at least several hundreds times and look at the potentially influencing factors. With the absent of the additional experiments, we decided to describe the whole distribution to a reader with the cumulative distribution function (CDF) and show that the frequency of the outliers in our 100-number distribution is rare. Figure 6.6 presents the CDF for the RTT values in the Tablet-to-PC scenario using IPsec and illustrates, for example, that the third quartile of this distribution equals to 2.60 ms and the 91.3% of the values do not exceed 3.00 ms.

The RTT time that we measured in the PC-to-Tablet scenario includes the transmission time of an ICMP ECHO_REQUEST message from the PC to the Tablet, processing time on both hosts and the latency of delivering an ICMP ECHO_RESPONSE back to the PC. The default size of an ICMP message equals 64 bytes (56 data bytes and 8 bytes of the ICMP header). When used with the IPsec BEET mode involved with HIP, the size of an ICMP message is augmented by the ESP headers and amounts 116 bytes. We measured the RTT time for a plain ICMP message of the size 64 bytes and 116 bytes, as well as for an ESP encapsulated ICMP

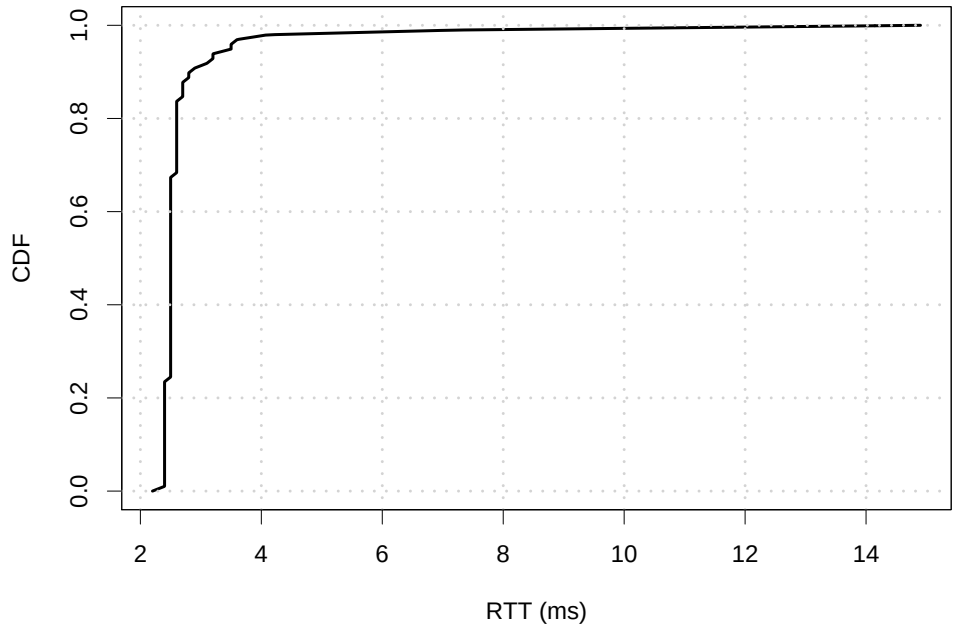


Figure 6.6: CDF for the RTT in the Tablet-to-PC scenario with IPsec.

packet (IPv6 over HIP). The results indicate that increasing the size of the ICMP packets merely does not affect the transmission latency. The major impact on the RTT in our experiments was observed in the case with IPsec (2.75 ms) that slowed down the packet processing on the Tablet by encapsulating the ICMP messages with ESP. Comparing to a plain IPv6 message, the IPsec BEET mode increased the median RTT value for the PC-to-Tablet connections by 0.67 ms. In contrast, the same value for the PC-to-Laptop scenario is only 0.13 ms. According to this comparison, the IPsec BEET mode involved with the Host Identity Protocol affects more seriously the lightweight devices than the ordinary PCs or laptops.

6.3.5 Throughput

IPsec ESP data encryption performed by the Tablet can reduce the maximum achievable throughput over the wireless link. We measured TCP throughput by an iperf tool generating TCP packets to a correspondent node. It is necessary to

Table 6.3: TCP throughput in different scenarios.

Throughput	**Mean ± Stdev (Mbps)**	
	Tablet→PC	Laptop→PC
TCP	4.86±0.28	21.77±0.23
TCP/HIP	3.27±0.08	21.16±0.18
TCP+WPA	4.84±0.05	–
TCP/HIP+WPA	3.14±0.03	–

mention that the WLAN AR introduces its own data encryption by means of the WPA protocol. Different tests had been performed to evaluate the overhead of the ESP and WPA data encryption. The average values of the throughput are presented in Table 6.3. An average value of 4.86 Mbps represents an upper bound of the throughput achievable by the Tablet acted as the Initiator (see Tablet-to-PC scenario). This value was measured with plain TCP/IP traffic in a totally open network with no encryption algorithms employed. Although the Tablet's specification claims supporting IEEE 802.11b/g standard with a maximum data rate of 54 Mbps, the Tablet's CPU or possibly bad device driver implementation impose their own constraints. Further analysing the results, we might conclude that the WPA encryption makes a minor impact on the throughput. Enabling the WPA access control on the WLAN AR reduces the data rate only by 20 Kbps. In contrast, the ESP influence is much stronger and reduces the throughput by 1.59 Mbps in the same network. The mutual impact of WPA and ESP is larger as double encryption is used.

In comparison with the Tablet, the Laptop achieves 21.77 Mbps of the TCP data rate over the same open wireless link (see Laptop-to-PC scenario). An interesting observation is that with the Laptop the impact of the ESP encryption involved with HIP is small as compared to the Tablet and equals 0.61 Mbps of the throughput decrease.

Figure 6.7 graphically depicts the results and shows the distribution of the TCP and TCP/HIP throughput over a WPA-free wireless link. The graph illustrates the influence of HIP on the TCP throughput as well as the difference in values achieved

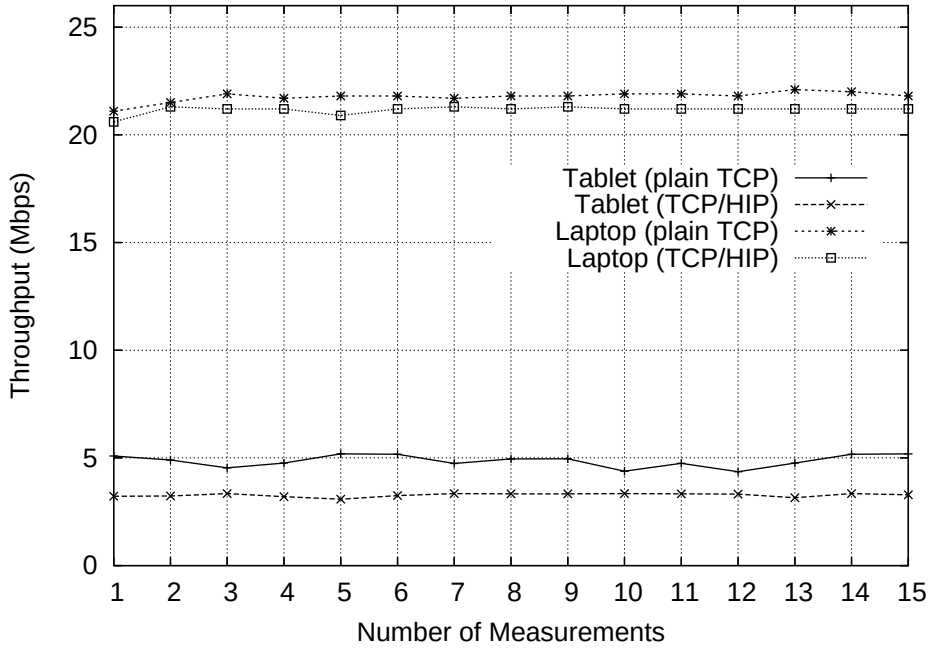


Figure 6.7: TCP throughput in an open wireless network.

by the lightweight Tablet and the much more powerful Laptop.

End-to-end security provided by HIP might be used not only for data protection itself but also for authentication to an access router as an alternative to the WPA algorithms in wireless networks. However, as the results above indicate for the devices with limited computational power, the data throughput and latency are notably affected by the ESP encryption in contrast to WPA encryption. In the absence of hardware-accelerated cryptography or in the case of its improper implementation, this might become a concern.

6.3.6 Duration of a mobility update

HIP sends mobility update packets when the IP address of a HIP mobile terminal changes. We measured the time to exchange three mobility update packets by manually changing the IP address of the Tablet's network interface to trigger a simple mobility event for HIP. We repeated our tests 35 times and calculated the average value, which equalled 287 ms (see Figure 6.8). This time was necessary to exchange three HIP mobility control packets between the Tablet and the PC

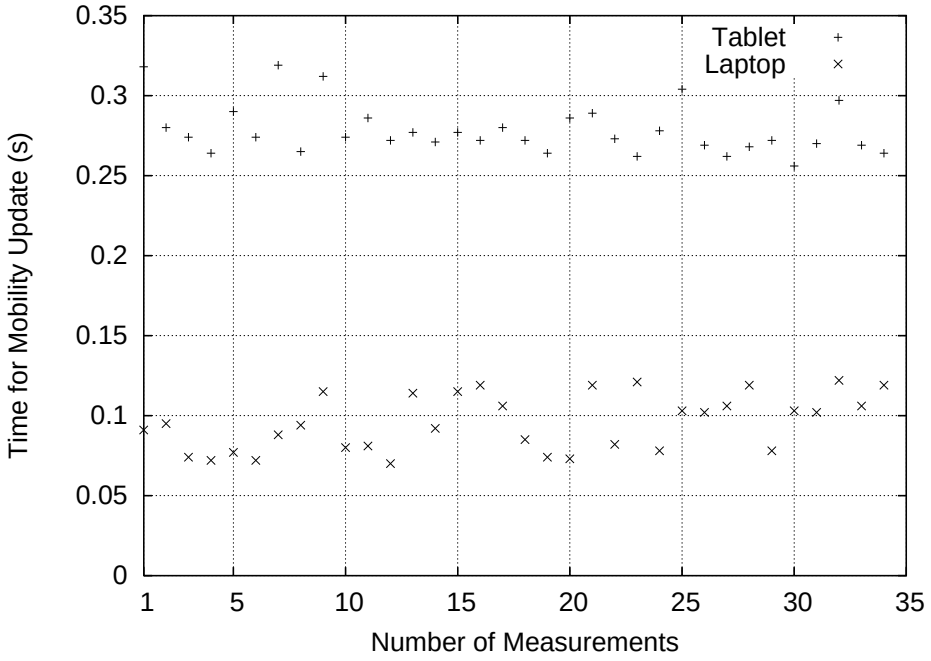


Figure 6.8: Duration of a HIP mobility update.

and it also includes the RTT multiplied by 1.5. Comparing to the Tablet, the 1.6-GHz Laptop was capable of completing the three-way mobility update with the correspondent node in 100 ms on the average. One reason for a notable deviation of the mobility update times visible on the graph can be the variation of the RTT observed in the previous experiments.

However, in reality the detected delay for a mobility update can be lower for applications. Once the correspondent node receives the first UPDATE packet it knows the Tablet's new location and can transmit data to the new address using the Credit-Based Authorization (CBA) mechanism. CBA limits the transmission rate to a new IP address until it is verified to be reachable by the last two UPDATE packets. Such practice prevents hijacking of arbitrary IP addresses. The average time for generating, sending and processing the first UPDATE packet in our experiment was around 20 ms.

Table 6.4: Power consumption by applications.

Application/Mode	**Current (A)**	**Power (W)**
HIP Base Exchange	0.36	1.33
ESP traffic (iperf with HIP)	0.38	1.41
Plain TCP (iperf without HIP)	0.38	1.41
Video stream from a server	> 0.50	1.85
Local video	0.27	0.99
Audio stream from a server	0.40-0.50	1.48-1.85
Local audio	0.20	0.74
Browsing (active WLAN)	0.35-0.50	1.30-1.85
Passive WLAN	0.12	0.45
Idle mode	0.12-0.14	0.45-0.52
Standby mode	< 0.01	0.04

6.3.7 Power consumption

Power consumption is a crucial issue for any portable device. The capacity of the Nokia 770's battery keeps the device in a standby mode for several days. However, the battery resources are exhausted quickly by the applications requiring data transmission over WLAN. The objective of measuring battery lifetime on the Tablet was to assess how expensive the Host Identity Protocol operations might be in terms of power consumption. We used an external multimeter to measure the consumption of the battery's current while the device was busy with various applications (see Table 6.4). Given the capacity of the battery and the current consumed by an application we were able to compute a theoretical time to deplete the battery. Alternatively, we ran the same application on the Tablet with a fully charged battery until its depletion to verify our empirical assumption about the lifetime. With HIP, the average current measured by the multimeter was 0.38 A. A fully charged 1500-mAh battery kept the Nokia Tablet working for about three and a half hours.

Our preliminary results show almost no difference in power consumption between

the HIP-enabled and non-HIP applications. Establishing a HIP association, mobility update as well as ESP encrypted traffic all consume a similar amount of the current (0.36-0.38 A) equivalent to a plain TCP/IP data connection. We interpret these results as caused by the low computational power of the Nokia's CPU which tries to utilize all available resources upon transmitting data over WLAN irrespective of the protocol and the application being used. However, we note that ESP does consume more power beside the non-ESP applications if compared to the data throughput. In other words, due to a lower bit rate caused by ESP data encryption a HIP application (using IPsec) would require a notably longer time for a similar task to be completed. For instance, taking into account our throughput measurement results the Tablet would be able to transmit 100 MB of data in 165 seconds over plain TCP/IP while the same task performed using IPsec would spend additional 80 seconds (totalling 245 seconds). In terms of power consumption the use of HIP and the accompanying IPsec would therefore intend a longer CPU utilization and consequently a larger amount of consumed energy.

6.4 Summary of the results

This chapter presented measurements and performance evaluation of the Host Identity Protocol on the Nokia 770 Internet Tablet. We found several interesting results on the use of asymmetric cryptography on lightweight Linux PDAs. The results are summarized below.

- In such scenarios where the Nokia 770 communicates through a single proxy server in the Internet, a HIP association establishment takes 1.4 seconds on the average (including two RTT of 2.5 ms). A three-way mobility update between the Nokia 770 and the proxy in this case lasts 287 ms on the average.

- For scenarios involving two mobile hosts or multiple parallel HIP associations, the delay for the end user increases almost twice. For two Tablets, a HIP association establishment takes 2.6 seconds on the average.

- For applications that do not require strong security (i.e., web surfing) the duration of the HIP association establishment with a server might be reduced by using a smaller DH modulus length. For instance, with the 768-bit DH Group and the 1024-bit RSA key length the average total BEX can be as low as 0.4 seconds (including two RTT of 2.5 ms).

- Surprisingly, the Tablet only achieves the data rate 4.86 Mbps on the average in a WLAN capable of 22 Mbps even without HIP. The use of the WPA encryption has negligible effect on the throughput, but the ESP encryption with HIP reduces the throughput to 3.27 Mbps on the average. It is still sufficient for most Tablet's applications.

- The RTT over WLAN is only several milliseconds in our experiments. The ESP encryption increases the RTT by less than one millisecond that does not noticeably affect the applications.

- The use of ESP encryption with HIP does not affect the instant current draw in the Tablet, although the energy cost per byte is higher with ESP due to reduced throughput. We noticed that the Tablet CPU is always fully utilized when an application transmits data over WLAN that depletes the battery in 3-4 hours.

- We consider our measurement results to be potentially applicable to other security and mobility protocols such as IKE [50] and MOBIKE [86], which rely on similar public-key and IPsec ESP operations.

7 Performance of Host Identity Protocol on Symbian OS

This chapter continues from Chapter 6 and reports on our experience with the Host Identity Protocol on another operating system, Symbian OS, and another class of devices, smartphones. While there are three open-source HIP implementations, little experience is available with running HIP on lightweight hardware such as a cellular phone. In this chapter we describe performance measurements of two different HIP implementations ported to Symbian OS. In particular, we compare *OpenHIP* and *HIPL* protocol implementations running on two Symbian S60 smartphones, Nokia E51 and Nokia N80. The chapter is largely based on our article *Performance of Host Identity Protocol on Symbian OS*[8] [53].

To check whether running IP-based security on smartphones is feasible, we performed HIP measurements over WLAN with the Nokia E51 and Nokia N80 in different scenarios. Particularly, we measured the duration of a HIP base exchange and its parts, as well as CPU load, RAM utilization and power consumption during several phases of HIP daemon work. We found that, e.g., with 1024-bit keys, a HIP base exchange with a server varies from 1.68 to 3.31 seconds depending on whether the mobile phone is in *standby* or *active* state respectively. Extensive analysis of HIP performance results allowed us to make conclusions and recommendations on using unmodified HIP on lightweight cellular phones.

Symbian OS is one of the leading operating systems for smartphones. In Q2 2008, 19.6 million Symbian mobile phones have been shipped globally. The amount increased by 5% from the same period of 2007 [4]. Smartphones in addition to traditional call and messaging functionality comprise a set of rich media applications making them similar to PCs in functionality. However, performance and usability of mobile applications are still a big concern. This is especially true with technologies initially designed to run on PCs. The contribution of this part of our work is evaluation of applicability of existing IP security and mobility solutions for smartphones. In addition, by developing Symbian ports of two HIP implementations we also contribute to the deployment of HIP. Our porting experience might be useful for those planning to bring open source software to Symbian OS.

The rest of this chapter is structured as follows. Section 7.1 describes our ports of HIPL and OpenHIP implementations. In Section 7.2 we present our experimental

[8] © 2009 IEEE. Used with permission.

network testbed followed by description of scenarios and measurement tools in Section 7.3. Section 7.4 contains selected measurement results of the base protocol and their in-depth analysis. Section 7.5 concludes the chapter with a summary of key findings and conclusions.

7.1 Main porting stages to Symbian

In this section we describe the key parts of the HIPL and OpenHIP porting process and challenges that we faced while migrating to the Symbian platform. We also present limitations of our prototypes. The porting process comprised several stages such as choosing the development environment, examination of the existing HIPL and OpenHIP source code, preparation of the Symbian project structure and *makefiles*, compilation, debugging and testing.

7.1.1 Development environment

We started with no prior knowledge and experience of Symbian OS. To begin porting process we needed to install an S60 3rd Edition Platform SDK for Symbian OS, a Carbide.c++ IDE and an Open C SDK plug-in for S60 3rd Edition SDK. The Open C plug-in brings support of nine standard POSIX and middleware C libraries to Symbian OS and allows easier porting of the existing C applications to S60 3rd Edition devices [27]. The availability of Open C plug-in played an essential role in our project as it provided access to many standard C functions and allowed to reuse the existing HIP implementations avoiding extensive modifications. Without support of POSIX C libraries it would not be feasible to port the project written in C without rewriting the major part of the code using the Symbian's native C++ programming language.

7.1.2 Project preparation

Before actual porting it is necessary to study existing software, its features and dependencies, and identify potential limitations of the target platform. To import the HIPL and OpenHIP code to the Carbide IDE and start working on the project we created a set of Symbian project files, *bld.inf* and *mmp*, which are platform and compiler independent files in Symbian OS. To create these files we studied existing Linux makefiles in the HIP projects. An mmp file contains all necessary information needed to build a component or a project. Application type (e.g., dll, exe), source

files, include directories, libraries, preprocessor macros, compiler and linker settings, stack and heap size, program capabilities and many other options are specified in the project definitions files mmp. A bld.inf file in turn comprises information about project mmp files, exports, and build platforms (e.g., WINSCW, GCCE). Having prepared the project files, one can build the project for different Symbian platforms and compilers.

For OpenHIP we chose a set of source files needed to run HIP in userspace mode, since we believed that this mode should be compatible with any platform that supports standard POSIX C libraries. We also included implementation of security association database (SADB) and PFKEY [68] protocol (with BEET mode support) for communication with SADB.

7.1.3 Compilation

The most common cause for compilation errors in the code was implicit data type conversions. Symbian compiler needs an explicit type casting to be performed. Furthermore, the Symbian compiler does not allow declaration of data types in the middle of a function. To avoid the compilation errors we had to add a number of extra definitions to the Open C header file *netinet6/in6.h* for HIPL project.

OpenHIP architecture, in turn, was better suited for porting. In fact, we did not change any system headers. Similarly with HIPL, we have been using preprocessor logical statements to separate system-specific code parts, and in case of missing functionality reimplemented it.

7.1.4 Debugging

When debugging the HIPL code we found a number of porting issues that arose only during execution of the HIPL daemon. The errors were caused by a difference in Linux and Symbian emulator compilers. The most interesting issues were detected in data structures that contain an array of zero elements. The first error type concerned the size of such structures. In Linux, a structure member declared as an array of zero elements (e.g., *uint8_t data[0]*) does not increase the size of whole structure. On the contrary, the size of the same structure in the Symbian emulator was bigger due to the size of the ”null” array treated by the Symbian emulator compiler as one byte.

The second error type was related to memory alignment. Upon referencing arrays of zero elements in a structure, the program running on Linux and on Symbian

emulator tried to access different memory blocks within that structure. Interestingly, we found that the Symbian compiler always rises the total size of the structure elements preceding a "null" array to an even value by adding an extra memory byte. As a result, to access a correct value recorded in the "null" array we had to shift the pointer appropriately. The piece of the code below illustrates this issue with a particular example. Here, the size of the structure members *group_id* and *pub_len* is 3 bytes, and Symbian compiler adds an additional 8 bits of memory prior to the member *public_value*.

```
struct hip_dh_public_value {
        uint8_t group_id;
        uint16_t pub_len;
        uint8_t public_value[0];
} __attribute__ ((packed));
```

It is worth mentioning that this specific feature has been detected only with the *mwcsym2* compiler that is used with the Symbian emulator. When the HIPL code was built for the target hardware with the *GCCE compiler*, the program behaved similarly as on Linux and all the changes we have made for the emulator needed to be restored. We did not find an explanation of such a difference between the compilers in the technical documentation.

7.1.5 Limitations of the prototypes

Both HIP implementations are entirely written in C and consist of a HIP userspace daemon and several HIP libraries. As the HIPL project was originally developed for Linux, the implementation contained several platform-dependent features such as the *NETLINK* socket for kernel and userspace communication. To protect payload data, HIPL uses the IPsec protocol that resides in Linux kernel. Due to limited public Symbian SDK and restricted access to Symbian network stack, our HIP prototypes for Symbian support only the base protocol part without ESP encapsulation of data packets in the system kernel. However, with OpenHIP we ported a userspace alternative – PFKEY protocol and SADB. As a result, we were able to successfully encrypt/decrypt UDP encapsulated incoming ESP packets.

Open C plug-in itself has a set of limitations that required us to modify the existing source code and disable a part of its functionality. Examples of unsupported or restricted features in Open C are *signals*, *fork()* and *exec()*, *wait()* and *waitpid()*

Table 7.1: Technical specifications of tested phone models.

Smartphone Models →	**E60**	**N80**	**E51**
CPU Clock Rate, MHz	220	220	369
SDRAM / Free Exec RAM, MB	64/21	64/18	96/50
Battery Capacity, mAh	1020	860	1050

functions, multiple *I/O consoles* [26]. Because of Open C constraints our HIP ports for Symbian use only UDP sockets to send HIP control packets excluding a raw-socket alternative as in the original HIP software. In OpenHIP, we used UDP encapsulation also for ESP packets, so that the raw socket limitation can be bypassed for ESP as well. The architecture of OpenHIP allowed us to support almost full-featured HIP implementation on Symbian OS. However, to run legacy applications over HIP an equivalent of Linux TUN/TAP driver needs to be implemented.

The HIP code ported to Symbian OS requires *NetworkService* system capability, which identifies a functionality for remote access to services that can produce cost to the phone user, such as network usage. Both HIP implementations compiled for the target hardware were signed with enabled *NetworkService* capability against a specific phone International Mobile Equipment Identity (IMEI) number and, without recompiling, cannot be used on any other S60 3rd edition mobile phone. To install the HIP daemon on another phone one has to sign the package with its own IMEI number at *www.symbiansigned.com*.

7.2 Our testbed

We have tested HIPL and OpenHIP code running on several Symbian phones: Nokia E60, Nokia N80, and Nokia E51. The first two devices are based on S60 3rd Edition platform and Symbian OS v9.1, whereas Nokia E51 is a slightly newer and more powerful smartphone that runs Symbian OS v9.2 and uses S60 3rd Edition Feature Pack 1 developer platform.

The general specifications of the tested phone models are summarized in Table 7.1. Nokia E60 has equivalent to N80 hardware resources and shows similar performance.

To obtain competitive results we measured HIP performance on a more powerful Nokia E51 phone equipped with a 369-MHz ARM11 CPU and a notably bigger RAM module of 96 MB. All phone models support IEEE 802.11 b/g connectivity standards with WPA2 encryption and a number of cellular standards including WCDMA, EGPRS, and HSCSD. Battery capacity in all phone models we used varies from 860 to 1050 mAh.

We measured the performance of HIP over WLAN. Our experimental network consisted of a D-Link DGL-4300 access router, three mobile phones, and an Intel(R) Xeon(TM) server with a 3.2-GHz CPU and 2 GB of RAM. The server was placed into the same network as cellular phones. The experimental testbed was similar to the one presented in Figure 6.1 except that Nokia Internet Tablets were replaced with Symbian smartphones.

7.3 Scenarios and tools

In basic scenarios, we established a HIP association between each of the Nokia phones and the server. We evaluated each stage of the base protocol separately including HIP daemon initialization, asymmetric key pair creation, daemon idle time, and protocol handshake (HIP base exchange). In the thesis we mainly report results obtained on Nokia E51 that showed better performance than two other models. Where possible we refer to the respective performance metrics measured on Nokia N80 and E60.

With Nokia E51, we utilized Nokia Energy Profiler, a convenient tool that runs on the phone in background and allows monitoring hardware usage in real time, as well as exporting data to a PC for future analysis. Profiling data includes information about such parameters as power and memory consumption, and CPU load.

With Nokia E60 and N80, we used Carbide.c++ Performance Investigator to collect and analyse data about usage of different resources. Performance Investigator consist of a profiler that gathers profiling data to a file during application runtime, and an analyser that runs on a PC and handles profiling data. Profiling data includes information about processes, threads, binary load, memory and power consumption.

7.4 Performance evaluation

This section presents and analyses the results of our measurements with the Host Identity Protocol on Symbian OS obtained with HIPL and OpenHIP prototypes.

Table 7.2: Base exchange duration with HIPL and OpenHIP.

Nokia E51	**Median/Average±Stdev (sec)**	
↓ **Scenario/Implementation** →	HIPL	OpenHIP
Phone→Server (Active)	3.21/3.17±0.11	3.05/3.09±0.17
Phone→Server (Standby)	1.66/1.68±0.06	1.93/1.90±0.12
Server→Phone (Active)	3.34/3.31±0.10	2.74/2.76±0.11
Server→Phone (Standby)	1.73/1.76±0.14	1.84/1.85±0.07
Phone→Phone (Active)	6.71/6.42±0.71	4.30/4.30±0.07
Phone→Phone (Standby)	3.83/3.78±0.13	3.49/3.50±0.12

7.4.1 HIP base exchange duration

In this section, we analyse HIP handshake duration in different scenarios. Surprisingly, we found a significant difference in HIP base exchange performance measured in *active* and *standby* phone states. We use terminology from [25] and slightly modify it. We call a phone state *active* when its display is switched on and refreshing (with backlight either on or off). In turn, we call a phone state *standby* when its display is in partial refresh (backlight is off; either date and time, text or animation is shown).

As Table 7.2 indicates, the total average time for HIPL base exchange initiated from the E51 phone to the server equals 3.17 seconds in the active phone state. Switching the phone to the standby mode reduces HIP base exchange duration almost twice (1.68 seconds). We believe the reason for such a great difference in performance is that in the standby mode no graphics are drawn and display is not refreshing, which releases extra CPU cycles that are utilized by the HIP daemon. On the other hand, in the active phone state, the processor load is close to 100% due to constant display refreshing, which prolongs processing time by the HIP daemon; we observed such behaviour by activating and deactivating the phone screen with running Nokia Energy Profiler.

The scenario with E51 as a HIP initiator is more natural for the Internet where most of the connections are initiated from mobile clients to servers. In the opposite direc-

tion (server → phone) duration of the HIPL handshake slightly rises and becomes 3.31 seconds in the active and 1.76 seconds in the standby state (see Table 7.2). Thus, time variation of the HIP base exchange performed in two opposite directions is insignificant. In our previous study [54] we obtained similar performance results with merely equal HIP handshake duration measured in two directions between a Nokia 770 Internet Tablet and a server. Further comparison to our previous work [54] indicates that the HIP base exchange performance on Linux-based Nokia 770 Internet Tablet is better than on Symbian-based Nokia E51 smartphone (1.40 versus 1.68 seconds), although the latter has more CPU and RAM resources. We explain this phenomenon as an impact of the Open C plug-in that is used with our Symbian HIP ports to wrap C function calls to native Symbian APIs.

Further analysis of the results in Table 7.2 indicates that the OpenHIP implementation shows slightly better performance than HIPL in the active phone state establishing a HIP association between the E51 and the server on the average during 3.09 seconds and in the opposite direction in 2.76 seconds. Though the effect of standby state in OpenHIP case is less significant than with HIPL (OpenHIP 35% against HIPL 47% time decrease when switching to standby mode).

A large part of the Internet traffic nowadays is generated by P2P applications. Keeping that in mind we measured HIP implementation performance running on two mobile phones. Our preliminary tests show (see Table 7.2) that two Nokia E51 smartphones in standby state are able to establish a HIPL association in 3.78 seconds and an OpenHIP association in 3.50 seconds. Switching to the active mode significantly increases the HIP handshake time for HIPL (6.42 seconds) while not seriously affecting OpenHIP (4.30 seconds).

Although it is interesting to know the HIP base protocol performance level in the standby phone state (i.e., when the HIP daemon is implemented as an engine and runs in background) we have to rely on the results obtained in the active state. This is because we expect user to interact with mobile phone (thus, activating the display) while using applications that might benefit from HIP.

7.4.2 Asymmetric key pair creation

Table 7.3 includes the median duration of creating a public-private key pair of different size on the Nokia E51. The results indicate an exponential growth of the key pair generation time with increasing the key length. With the conventional 1024-bits keys, the median time to generate an asymmetric DSA key pair on E51 is 25.4 seconds. The generation of an equivalent RSA key pair is much faster with the

Table 7.3: Creation of a key pair of different size on the Nokia E51.

	Median time (sec)		
↓ Algorithm / Key (bits) →	*512*	*1024*	*2048*
DSA	4.9	25.4	232.1
RSA	0.52	3.7	27.1

median time 3.7 seconds. It is worth noting that the use of the keys with the length over 1024 bits) would produce a delay of almost four minutes with DSA and half a minute with RSA.

One might argue that keys are needed to be created only once, i.e., upon installing HIP and this would not affect the overall phone performance in the long run. Nevertheless, we find stressing of a mobile phone even for a short period to be inconvenient and slowing down the phone's operations when its normal functionality is crucial (as with emergency calls). According to our practical experience, the generation and usage of lengthy keys on mobile phones with lower amount of RAM and CPU power, such as E60 or N80, seriously affect the handset performance and can make the phone completely unresponsive for several minutes.

Long generation time of a public-private key pair on the Nokia E51 illustrates the necessity to look for different approaches for managing the keys on mobile phones. One approach can be to generate the keys on a PC and securely transfer them to a phone. This can be performed by the phone's user assisted by an application either supplied with the mobile phone or available elsewhere. Another approach can be to precreate the keys for a mobile phone and transfer them to the device before it is sold. In this case, any additional user actions are avoided. Further details on the key distribution on mobile devices are left outside of this work.

7.4.3 CPU load

In this and the following subsections we report on the indicators of hardware utilization that were collected with the Nokia Energy Profiler on the Nokia E51.

The CPU Load during the HIPL daemon initialization and asymmetric key pair

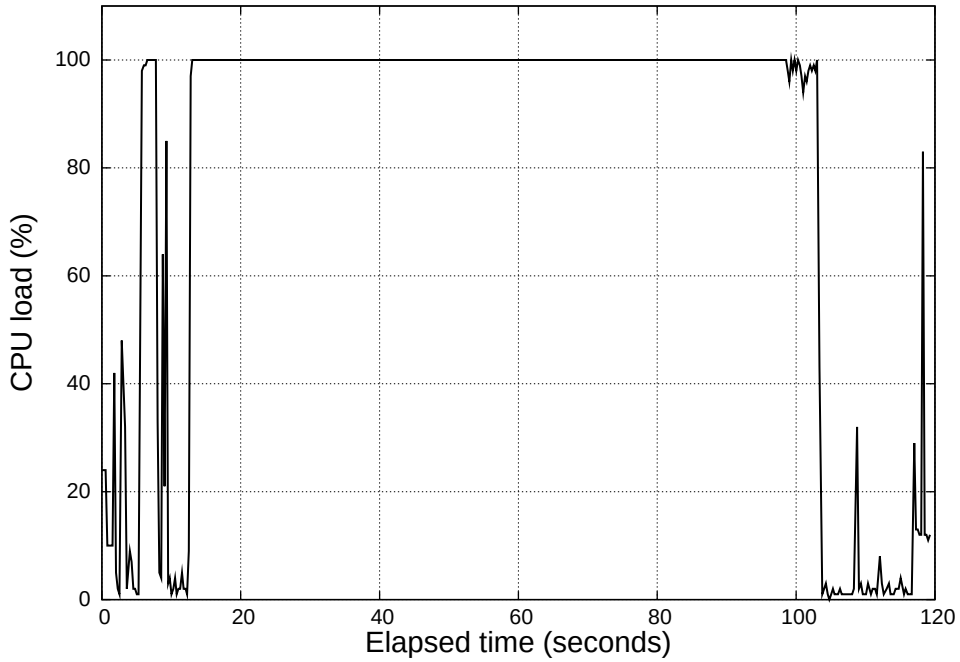


Figure 7.1: HIPL daemon initialization. CPU load on E51.

creation on E51 is presented in Figure 7.1. Most of the 2-minute measurement period the CPU load stays at 100% and this corresponds to the daemon initialization (up to the 12th-second timestamp) and the generation of four different public-private key pairs (the interval from 12th to the 104th second). The rest of the graph (from the 104th second onwards) has several peaks that account for precreation of the HIP R1 packets. In idle time the HIPL daemon does not consume much of processor power. We also noticed that switching the phone to the active state significantly rises the CPU load. The CPU utilization with the OpenHIP implementation is similar to the HIPL case.

7.4.4 RAM usage

Although each mobile phone has certain amount of RAM memory, only a part of it is available to applications. For example, on the Nokia E60 only 21 out of 64 MB are available to the executables. The rest of the memory is reserved for the exclusive use by the system. This reduces the number of the user applications that can be simultaneously ran on the device. According to our profiler data, the memory usage

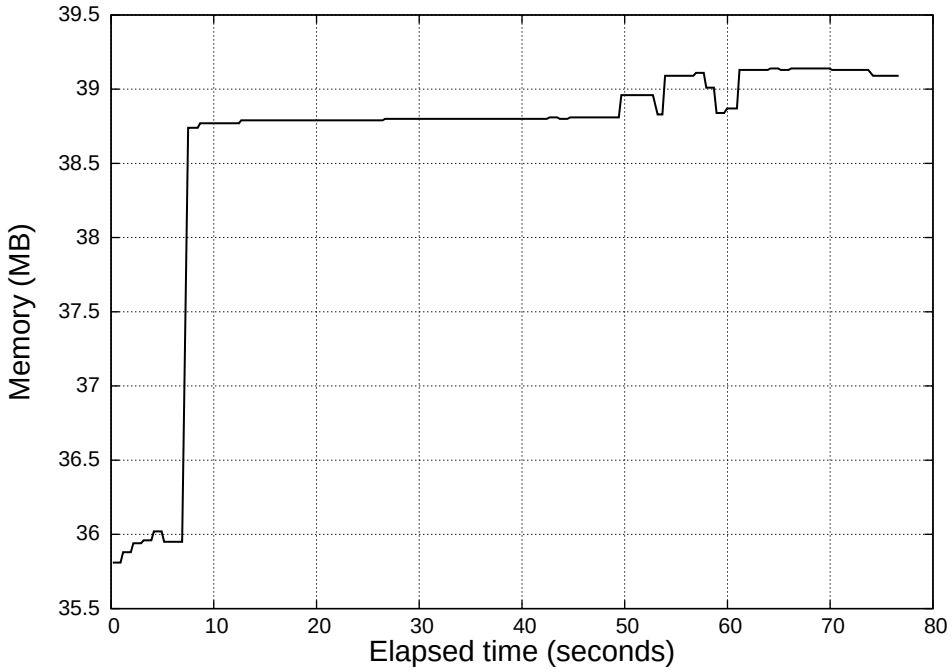


Figure 7.2: OpenHIP daemon initialization with BEX. RAM usage on E51.

on the N80 phone stays on the level of 20 MB during the HIPL daemon initialization, the key generation and the HIP handshake. Note that this value depicts the overall memory use by all running applications. Assuming that other applications are in the idle state, we can figure out the memory use by the HIP daemon. On the Nokia E51, the memory use dynamics are almost the same as on N80. The only difference is the amount of available RAM, which is larger on E51. According to the technical specifications, E51 allocates to the applications approximately 50 MB out of the total 96 MB of RAM.

Figure 7.2 depicts the RAM usage by the OpenHIP implementation on the Nokia E51 during the daemon initialization and the BEX. The reader should take into consideration the fact from the previous paragraph and notice that the graph shows the overall memory consumption by all applications. In fact, HIP starts its initialization at the point of approximately 36 MB. The time interval from the 8th to the 50th second corresponds to the key creation and serialization. Since OpenHIP stores all precreated keys in RAM (as well as serialized to the file system), the memory use increases by 3 MB. However, later during the base exchange and the idle time (the time interval from the 50th to the 80th second), the memory usage

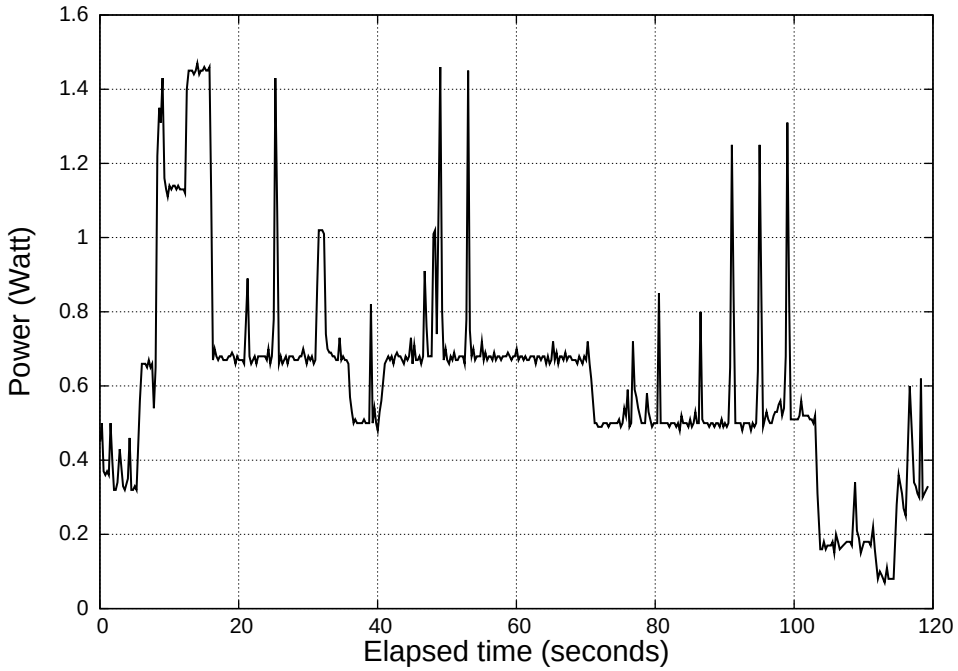


Figure 7.3: HIPL daemon initialization. Power consumption on E51.

does not grow drastically but only increases by half a MB during the processing of the BEX packets (which is freed afterwards) and adding the SAs to the SADB. In fact, the total RAM usage of 3 to 4 MB by the protocol is within the normal bounds and should not stress the performance of a mobile phone. Hence, from the RAM utilization prospective both HIPL and OpenHIP could be run on a Symbian mobile device without major changes to its architecture.

7.4.5 Power consumption

Figure 7.3 illustrates the power consumed by the E51 phone with the running HIP daemon during its initialization and idle time. The value of 0.62 Watt represents the average consumed power over a 2-minutes measurement period. The peaks on the graph between the 8th second and the 53nd second timestamps show the maximum power consumption over the measurement period and they account for the creation of two DSA and two RSA key pairs.

To compare how HIP affects the battery life of the phone we measured the average power consumption while the phone was in ”normal” use (i.e., no HIP daemon was

running) and with the HIP daemon doing the base exchange. As a result, we obtained 222 mW/60 mA and 333 mW/90 mA on the average respectively, which after the extrapolation corresponds to 18 and 12 hours of a 1050-mAh battery lifetime. In reality, the constant exchange of the HIP control packets is abnormal behaviour of the HIP daemon, so the purpose of these results is rather to illustrate an instant power consumption by the cryptography operations constituting the BEX.

7.5 Summary of the results

This chapter presented measurements and performance evaluation of two separate Host Identity Protocol implementations on Symbian OS. Below we summarize the most interesting and important results that at the same time can serve as recommendations on the use of public-key cryptography on lightweight Symbian OS mobile phones.

- A single HIP base exchange between the Nokia E51 and a proxy server lasts for 1.7-3.2 seconds on the average depending on the phone state (active or standby).
- In turn, two Nokia E51 require 3.5-6.4 seconds on the average to establish a HIP association.
- The public-private key pair generation might stress the phone and make it unresponsive for up to four minutes, especially with the key length greater than 1024 bits. However, this issue can be addressed by several alternative approaches such as predistributing the keys before the device is sold or generating the keys on a PC and transferring them to the mobile phone.
- Key creation boosts the CPU utilization and consumes a notable amount of power but otherwise the HIP daemon in the idle state consumes few resources. The impact of the WLAN transmission on energy consumption has to be considered separately.
- The OpenHIP implementation had been easier to port and showed slightly better performance over HIPL.
- Better performance results could have been achieved if HIP was implemented using the native Symbian C++ APIs rather than the Open C plug-in. This is because Open C is a wrapper to the native Symbian APIs and, thus, produces

an additional overhead comparing to the native applications. However, it should be noted that this remark is rather based on common sense, and we cannot provide any numbers illustrating the level of the potential performance increase.

8 Security and Mobility in Wireless LANs

In the previous chapters we evaluated performance of the Host Identity Protocol for two types of the resource-constrained mobile platforms (a Linux PDA and a Symbian smartphone). In this chapter, we present a more general view on similar problems of security and mobility, and expand our analysis and evaluation from the end user devices towards an ecosystem, where communications of such devices with other components of the infrastructure need to be secure. In particular, we consider WLAN networks, which have recently become a common way to access the Internet, and address the following issues: authentication and access control, host mobility and multihoming, communication security and prevention of several types of external attacks on the operator's infrastructure.

In the following sections, we first highlight several essential bottlenecks of WLANs and explain our motivation and then introduce our own distributed authentication architecture intended to prevent wireless networks from unauthenticated access, impersonation and network abuse, data interception and different types of attacks. In addition, we evaluate performance of selected architectural components based on the measurements in a test network. The chapter is mainly based on the article *Distributed User Authentication in Wireless LANs*[9] [61].

Our architecture utilizes several benefits of the Host Identity Protocol. It integrates an operator-specific HIP proxy with a HIP-aware firewall running on each of the operator's WLAN access routers (ARs) so that the mobile clients can instantly gain WLAN access and move freely within the operator's network. To build our architecture, we have implemented a port of the HIPL protocol implementation to run on OpenWrt WLAN ARs. We analyse measurement results obtained on two types of ARs with highly varying resources, La Fonera FON2100 and Gateworks Avila GW2348-4. Performance evaluation suggests that a two-level approach consisting of a single HIP proxy server and a distributed HIP firewall is appropriate, given limited hardware resources of some WLAN ARs. The presented architecture is planned to be deployed in panOULU [3], a public city WLAN in Finland.

The rest of this chapter is structured as follows. Section 8.1 discusses the issues with existing wireless networks and states the objectives we aim to achieve with our architecture. Section 8.2 presents our distributed WLAN authentication architecture. In Section 8.3 we describe our port of the HIPL implementation to OpenWrt platform and experimental testbed used for our measurements. Section 8.4 contains the

[9] © 2009 IEEE. Used with permission.

measurement results of CPU and memory utilization on two different AR models. Section 8.5 concludes the chapter.

8.1 Motivation

An increasing number of laptop and smartphone users utilize WLANs for Internet access at work, home, and public places. Unfortunately, authentication mechanisms in WLANs remain cumbersome, unreliable and disruptive to users. Typically, the WLAN users are required to re-enter their login and password periodically through a captive web page, or manually configure the WPA keys or 802.1X settings. The owners of open WLANs risk to fall under police investigation in case of network misuse.

Open wireless networks (such as presented in Figure 8.1) usually have no mechanisms for access control, protection of data integrity and confidentiality. The core of the problem is that anyone can gain access to the network without providing and validating their identity. This allows an attacker to perform illegal actions and potentially cause damage to the infrastructure without being caught. On the other hand, publicly available WLANs usually use no encryption, hence all the traffic transmitted over the air can be easily sniffed, analysed and used for malicious purposes. To eliminate such risks, we need the mechanisms that would provide reliable data protection and access control.

Several trends make the situation harder with time. Some emerging devices, such as smart key chains, are being equipped with WLAN capability, although missing a screen to display and enter login information. Furthermore, recent advances in breaking WPA encryption [10] and 802.1X [11], necessitate to look for far robust IP-layer encryption over the wireless link.

The above issues have been addressed in a number of research projects resulting in several potential solutions (e.g., [58, 8]). However, none of the methods achieves all of the following properties at the same time: 1) disruption-free user authentication 2) protection of operator's infrastructure from external attacks 3) host mobility and multihoming 4) IPsec encryption over the wireless link.

[10] Researchers Crack WPA Wi-Fi Encryption, `http://it.slashdot.org/article.pl?sid=08/11/06/1546245`

[11] 802.1X vulnerabilities, `http://www.microsoft.com/technet/community/columns/secmgmt/sm0805.mspx`

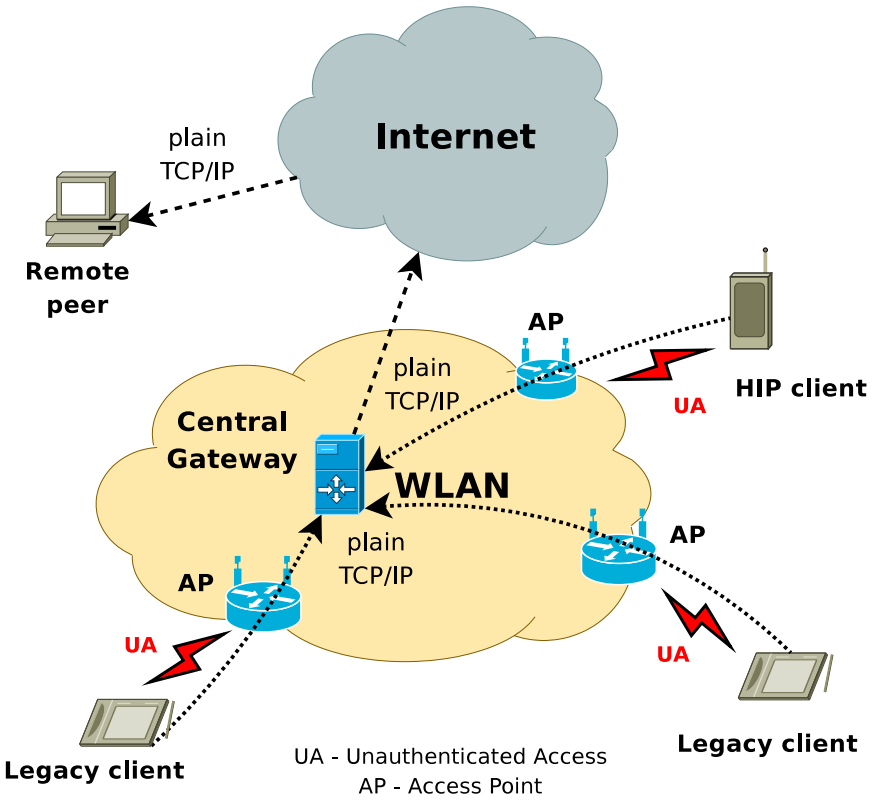


Figure 8.1: Open network access model.

8.2 Distributed authentication architecture

In this section we present our approach for automatic WLANs authentication according to the design goals stated in the previous section. We start by describing the main architectural components and principles, proceed with discussing the approaches for synchronization of the distributed firewalls, then highlight several ideas on the rule management and finally mention a number of potential methods for subscribing to the service.

8.2.1 Architectural components and principles

The general view of our architecture is shown in Figure 8.2. We propose to utilize HIP as a backbone that supports client mobility and multihoming in addition to WLAN authentication. A HIP-enabled mobile client establishes a secure association

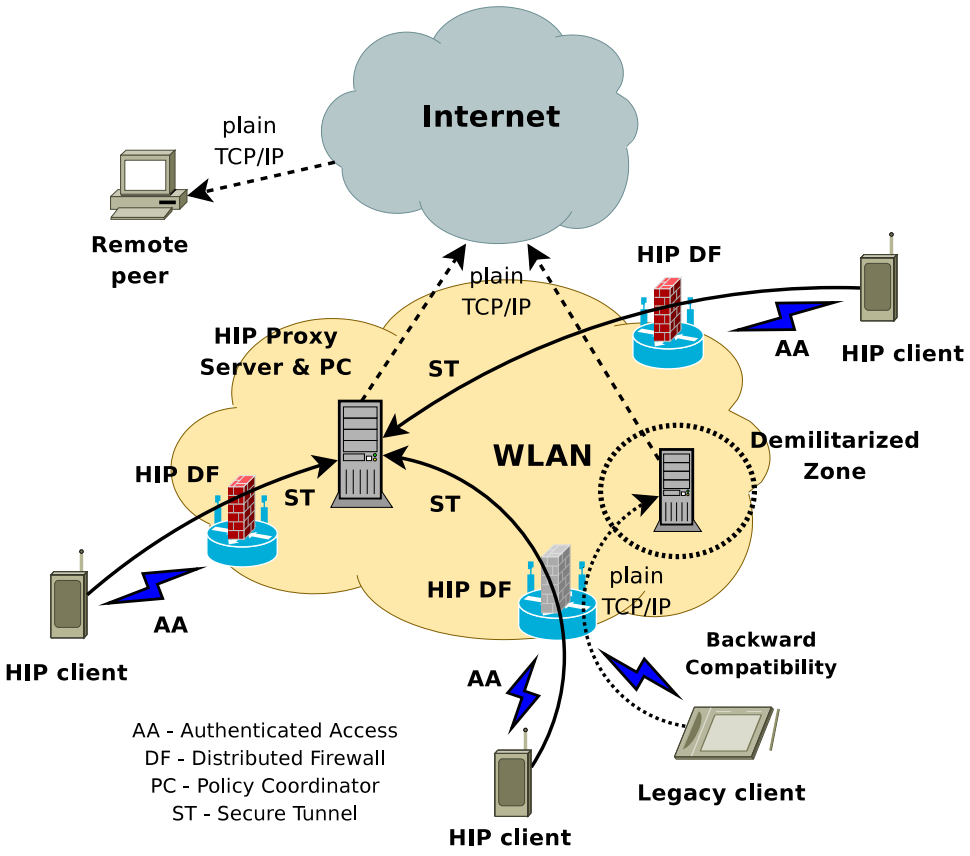


Figure 8.2: Distributed authentication model.

with a central HIP proxy installed on the default gateway in the core network. User data is then protected by the ESP secure tunnel. HIP and IPsec associations are updated when the client moves to another location within the network. Another role of the central HIP proxy is to enable connections from the mobile HIP clients to the remote servers in the Internet that do not understand HIP. In a simple scenario, the HIP clients connect to the non-HIP Web servers through an HTTP/HIP proxy to secure their browsing sessions.

To solve the authentication problem, we introduce a set of interconnected HIP-aware firewalls called the *distributed firewall* (see Figure 8.2). The main purpose of a HIP-aware firewall is to filter traffic based on a predefined list of allowed Host Identity Tags (HITs) that authenticate clients to the operator. Additionally, the firewall can perform a digital signature check (as, for instance, in PISA project [37]).

Checking the signatures provides a higher authentication level but involves the cryptographic operations. The overhead can negatively affect the overall data throughput of the firewall and reduce the number of clients served with a reasonable QoS level.

HIT-based filtering can be sufficient for a WLAN network when a client connects to the Internet through the central HIP proxy. Even if an attacker is able to generate a valid HIT, it would fail to complete the HIP base exchange after receiving an R1 packet (due to lack of knowledge of a private key). After a HIP association is established, ESP traffic is filtered using SPI values stored from the HIP base exchange.

However, in a scenario where a mobile client communicates with another mobile client in the same WLAN network, an attacker has chances to replay the HIP control and the ESP packets (sent between two mobile hosts) and establish a communication with another attacker within the same network, thus using the network resources on behalf of the legitimate clients. To eliminate such risks we suggest to use an extension for client authentication and authorization at the middleboxes proposed by Heer *et al.* [36]. In our architecture, the Wi-Fi ARs would play the role of the middleboxes that can authenticate the HIP and ESP packets transmitted between two mobile clients in the same WLAN.

8.2.2 Synchronization of firewalls

Distributed HIP firewalls are placed on the edge of the wireless network and perform packet filtering based on the predefined access control list (ACL). In other words, traffic from a registered HIT can successfully pass through the firewall and flow to the core network. In such an architecture, all participating Wi-Fi ARs (HIP-aware firewalls) should maintain a fresh copy of the rules, so that a newly registered customer can pass authentication successfully anywhere within the WLAN coverage.

Synchronization of the ACLs needs to be efficient. The lists should be updated frequently without flooding the network with signaling traffic. In this paper we are not proposing any specific protocol for synchronization but offer a general architecture overview and suggest the following two approaches for synchronizing the ACLs between the firewalls:

- First, a firewall can store the complete ACL locally and query the central policy coordinator server on-demand (when no matching rule is found locally)

Algorithm 1 ACL synchronization algorithm.

Require: $certificate \neq NULL$
Require: $address_{server} \neq NULL$
  if $authenticate(certificate, address_{server}) = TRUE$ **then**
    $updateACL(address_{server})$
    $resetStatsForNewEntries()$
    $sortACLbyStats()$
  else
    $reportError()$
  end if

or at fixed intervals (this approach will cause a higher network load). A request will update the list of allowed HITs.

- Second, the AR firewalls can form a peer-to-peer network (for instance, using a DHT). Each AR would store a partial list of allowed HITs and perform on-demand queries to the overlay if the matching rule is not found locally.

For the first approach, we assume that a centralized policy coordinator is present in the network (in Figure 8.2 it is placed on the gateway). Such policy coordinator holds the current list of the allowed HITs (or a user registration database). A simple ACL synchronization protocol is exemplified in Algorithm 1.

To synchronize the ACLs, an AR is required to authenticate itself to the central server with a certificate, or using other available mechanisms. Upon successful registration, the AR merges the ACL with the new updates.

8.2.3 Rule management

With linear search, the packet filtering time on a Wi-Fi AR firewall depends on the position of the appropriate rule in the ACL. Classifying and matching a packet takes $\Theta(n)$ time in the worst and $O(1)$ in the best case, where n is the number of the rules in the ACL. Our initial experimental results with packet filtering time on Avila confirmed the need to employ a rule management technique to achieve better filtering performance than that provided by pure linear search.

A simple strategy to sort the rules in the ACL may involve collecting per-packet statistics and trying first the most frequent rules. An alternative solution can be a hash table that guarantees $O(1)$ search time. However, this will constrain the

flexibility of the rules and restrict the search criteria to only one key, e.g., the source HIT. Such approach might successfully work in a simple setup, but more flexible systems would require a more comprehensive algorithm. The hash table approach might be infeasible if the ACL controls the number of the remote servers the user is allowed to access. In this case, each rule in the ACL needs to have a destination HIT. However, a hash table cannot provide filtering based on multiple criteria. Yet another approach for matching the rules is to use tries and ternary trees. For our architecture, we do not specify any particular method for ordering and matching the rules. In fact, this is a general topic, which has been extensively studied in the literature [7, 30, 45, 99].

8.2.4 Service subscription

Prior to the first-time connection to an operator's WLAN, a client in our architecture is supposed to perform a registration or, in other words, to subscribe to the service. The registration has to be done in a secure way, resulting in authenticating the client to the operator and storing the mapping between the user identity and her HIT in a registration database. The registration database is then synchronized among all firewalls. In practice, there might be several alternative methods to accomplish this procedure, including registration in person at an office by providing an identity card; subscription to the service in the Internet using a banking authentication service; registration by mobile phone or via email. Each mechanism has its own advantages and disadvantages. More details on the different subscription methods can be found in the literature. For instance, Kuptsov and Gurtov [60] describe a simple web and email-based registration system. In general, the design of such systems needs to have a good trade-off between the security and the convenience of usage.

8.2.5 Compatibility with legacy clients

A large-scale deployment of our architecture can require a transition phase, when not all of the mobile clients will understand HIP. In this section, we consider two possible approaches to provide backward compatibility to the legacy clients in the early deployment stages. Both approaches require support of legacy and HIP-enabled clients in the WLAN ARs.

In the first approach, we can run a HIP proxy on a WLAN AR. This proxy would provide support for non-HIP (legacy) clients by translating the plain TCP/IP data to the HIP and ESP traffic. In this case, the WLAN AR would need to perform the

HIP base exchange and IPsec encryption (between the AR and the central network gateway). Our measurement results in Section 8.4.2 indicate that this approach is inefficient for a resource-constrained WLAN AR due to its computational overhead. It limits the serving capacity of the WLAN ARs and the scalability of the whole architecture.

A more rational approach to deal with the legacy clients is to perform a simple port switching on the Wi-Fi ARs and thus decrease the load on the infrastructure. As the use of the HIP proxy on the ARs does not deliver any additional benefits other than supporting the legacy clients, it makes sense to replace it with a port switching technique.

An example of such setup is illustrated in Figure 8.2. An AR routes the traffic from a HIP-enabled client towards the central HIP gateway establishing a secure tunnel and filtering the HIP and ESP packets on the HIP-aware AR firewall. At the same time, a legacy mobile client is routed by the same AR to a demilitarized network zone and can be served with a lower QoS level (depending on the network policy).

8.3 Experimental testbed

This section presents our experimental testbed. It starts with a description of the HIP on Linux (HIPL) [1] porting process and highlight the challenges that we confronted during migration to the OpenWrt platform. Next, we show the components of our network setup and introduce the status of the architecture deployment in the panOULU WLAN.

8.3.1 Porting HIPL to OpenWrt

We ported the HIPL implementation to two AR models, La Fonera FON2100 and Gateworks Avila GW2348-4, both running the OpenWrt Linux distribution.

Porting of HIPL to the OpenWrt platform was not a straightforward process and required efforts with both AR models. Among the problems we faced were various memory management issues, missing dependencies, and hardware constraints. We have chosen OpenWrt as a reference Linux distribution because of its wide range of supported hardware platforms. Fortunately, OpenWrt is known for its good support of FON and Gateworks boards. Another consideration was a large and growing community of developers that work on the OpenWrt project.

During the HIPL software migration to OpenWrt we detected several critical bugs that were hard to locate and eliminate. Besides that, we rewrote a number of parts of the HIPL code completely to avoid library dependencies. For instance, we needed to reimplement the list data structures to remove *glib* library dependencies. Interestingly, the HIPL code running on ARM processor (Avila) required static typecasting to the character pointer type for memory copy operations. Otherwise, we were getting ambiguous results; as an example, we have observed that copying of the *in6_addr* structure would copy correctly only 96 bits and fill the rest of the structure with zeroes.

As a summary, porting of the current HIPL implementation to other architectures supported by the OpenWrt platform should be feasible, but researchers can encounter problems related to a specific platform. Since HIPL is not included in the OpenWrt distributions, it should be compiled with an OpenWrt SDK and the HIPL patches that are publicly available.

8.3.2 Experimental setup

Our network setup (see Figure 8.2) comprised a set of the Wi-Fi ARs running a HIP-based distributed firewall. The first AR model we used was La Fonera FON2100 that has 16 MB of RAM, 8 MB of Flash memory, and a 32-bit MIPS processor running at 183 MHz. The second model, Gateworks Avila GW2348-4, is more powerful than the previous one, combining on an average-sized board 128 MB of RAM, 32 MB of Flash, and a 533-MHz Intel CPU.

Another component of the implemented architecture was a central HIP proxy server, a desktop-like PC, that acted as the main gateway for the whole WLAN network connecting it to the Internet. In addition, the testbed included a remote peer and a number of mobile clients ranging from a Nokia 810 Internet Tablet and a Symbian S60 smartphone to a mini-laptop ASUS Eee PC 901. There were both HIP-aware and non-HIP hosts among these mobile devices. The clients used two publicly available HIP implementations, HIPL and OpenHIP [2]. All components of our experimental testbed for distributed user authentication in a wireless network are illustrated in Figure 8.2.

8.3.3 Considerations for deployment

Our system works in a way that the HIP-enabled users establish an association with the central HIP proxy server by performing a HIP base exchange. Each packet

sent from a mobile client is filtered by the distributed firewall running on the HIP Wi-Fi ARs based on the source and destination HITs. Such scenario provides multiple benefits, including strong user authentication to the WLAN network and HIP terminal mobility. In addition, all transmitted data is protected by IPsec.

On the other hand, if there is a need for an incremental architecture deployment in a large public WLAN network, such as panOULU, our architecture can be easily extended to provide backward compatibility for the legacy clients. This can be realized by a simple port switching technique on the Wi-Fi ARs as described in Section 8.2.5.

Ideally, we recommend to deploy the complete architecture at once so that each WLAN user becomes HIP-aware and is able to authenticate itself to the network. We believe that universal client authentication would significantly reduce the probability of a network abuse. However, we admit that in practical situations for large operator environments an incremental deployment is more feasible. In such cases, the operator may provide backward compatibility for a certain transition period needed to install HIP on the legacy client terminals.

8.3.4 Deployment status in panOULU

Deployment of our authentication architecture in panOULU, a city-wide WLAN in Finland, is in its initial phase. We have installed an HTTP/HIP proxy on the main network gateway. The proxy allows the mobile clients understanding HIP to establish secure HIP associations with the central gateway. The proxy authenticates the clients to the network, provides terminal mobility and encrypts user data over an unprotected wireless link. Our preliminary tests showed that a mobile HIP user can successfully connect to the panOULU network, secure the browsing sessions by constructing an IPsec tunnel with the central HIP proxy server and keep the sessions ongoing while changing the network attachment point.

As the next step, we added a La Fonera FON2100 AR to panOULU with the running HIP firewall and proxy extensions. Initial experimentation indicated that limited hardware resources of this AR model are stressed by the HIP proxy component that, in turn, is not able to serve many clients simultaneously. In the future, based on the performance comparison of different Wi-Fi AR models, we plan to choose the most suitable hardware and continue deployment of our architecture in the panOULU network.

8.4 Performance evaluation

This section presents our initial measurement results on two Wi-Fi AR models with different hardware resources, La Fonera FON2100 (from now onwards *Fonera*) and Gateworks Avila GW2348-4 (from now onwards *Avila*). The results have been measured in two modes with each AR running as a HIP proxy and a HIP firewall.

HIP involves public-key cryptography and IPsec encryption that can easily stress lightweight resources of Wi-Fi ARs. One of our objectives was to evaluate the impact of such computationally-intensive operations on the performance of our authentication architecture. Our previous work [54, 53], which studied HIP performance on the Linux-based Nokia Internet Tablets and Symbian smartphones, served as a good reference for performance evaluation in this article. In addition, HIP has been evaluated on stationary Internet hosts with conventional PC-like resources [40, 38, 48, 87].

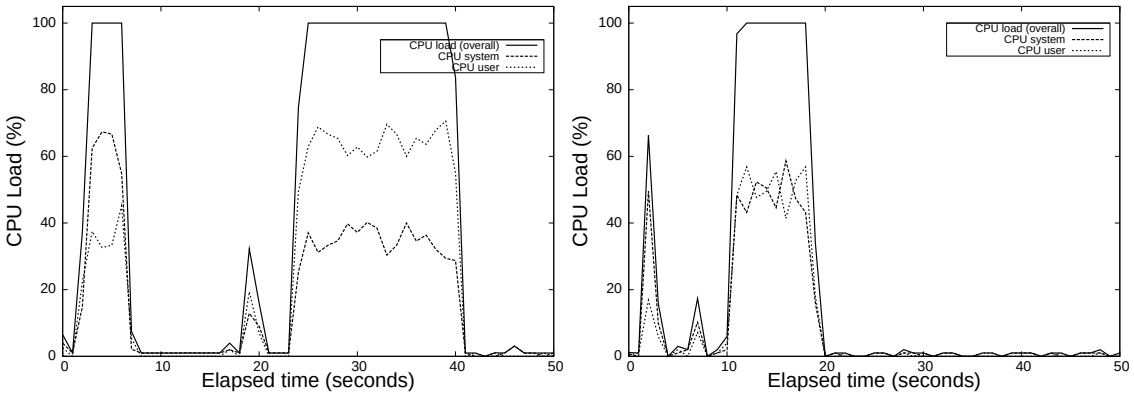
We have made an interesting observation that 100% CPU utilization does not necessarily indicate a performance issue for a particular mobile device. Rather, this can be interpreted as the utilization of all available resources by running applications when the system allocates maximum capacity to them. We have noticed that when all applications are in the *idle* state, the CPU utilization is about 1%. However, upon invoking a resource-demanding application, the system will release all available CPU cycles to it. On the other hand, with multiple applications running in parallel, the Linux scheduler guarantees no starvation for each task by fairly allocating the time slices.

8.4.1 Firewall mode

This section contains an analysis of our measurement results on Fonera and Avila running in the HIP firewall mode.

A HIP-enabled firewall on a Wi-Fi AR does not require running the HIP daemon, unless the HIP daemon itself is used for user registration or similar tasks. In our experiments, we ran only the HIP-enabled firewall as the crucial component in our architecture.

Figures 8.3a and 8.3b illustrate the copying task of a 30-MB file over SSH and HIP, while the traffic was being filtered on the Wi-Fi access routers in the middle. The ACL on the Wi-Fi AR contained four rules. The first peak on both graphs (the time interval from the first to the sixth second in Figure 8.3a; the time interval from the first to the fourth second in Figure 8.3b) corresponds to the HIP base exchange



(a) Fonera (b) Avila

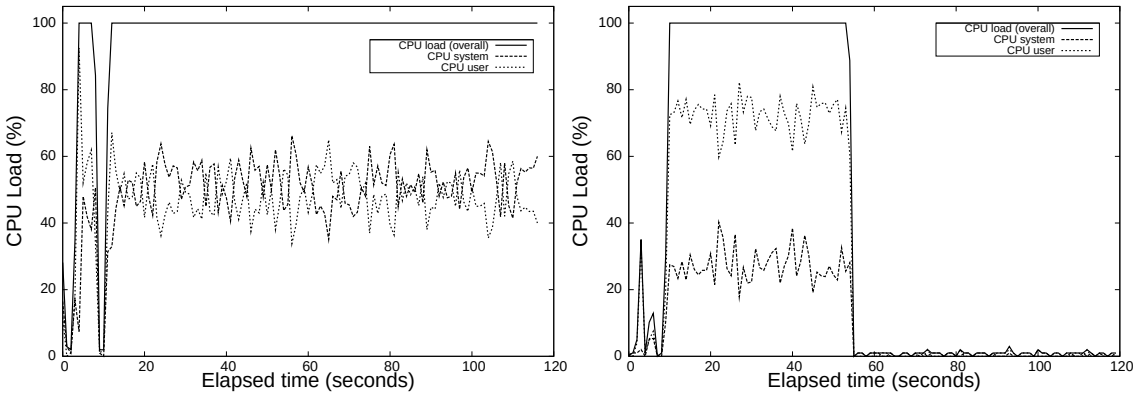
Figure 8.3: CPU load in the firewall mode.

between the mobile client and the default gateway in the network. The BEX packets were filtered by the firewall on the Wi-Fi AR based on HITs. The second peak on the graphs (the 19th second in Figure 8.3a; the seventh second in Figure 8.3b) accounts for the SSH key exchange between the mobile client and a remote peer. Finally, the interval from the 23rd to the 42nd second in Figure 8.3a and from the 10th to the 20th second in Figure 8.3b corresponds to the filtering of the ESP packets by the firewall. As the figures show, Avila significantly outperforms Fonera in terms of the time required to complete the whole task, spending in total twice as fewer seconds as Fonera (20 versus 42 seconds). This result indicates that faster AR hardware is necessary to provide sufficient performance of filtering operations in a distributed HIP-based firewall.

Our results on memory utilization in the firewall mode ensure a good level of performance on both AR models. We found that although only 1 MB of RAM is available after the firewall start-up on Fonera, it is sufficient to sustain two-three mobile clients. With Avila, the situation is better as only 21 MB of the total 128 MB of RAM are used on the average. Thus, the amount of RAM in the access router does not make a significant impact on the firewall performance.

8.4.2 Proxy mode

This section presents the results obtained on Fonera and Avila running in the HIP proxy mode. Figures 8.4a and 8.4b depict the CPU utilization during the bulk



(a) Fonera (b) Avila

Figure 8.4: CPU load in the proxy mode.

copy of the same file over SSH as in the previous experiment. Please note that the plots do not give the CPU load per application but instead for the whole system. Assuming that other applications are in the idle state, our scenario suggests that the dynamics of the CPU utilization on the graphs account for the HIP daemon and the proxy. For both experiments with Fonera and Avila the setup and the size of the transmitted file were identical.

The difference with previous test is the following. A mobile client in this experiment is HIP-unaware, hence the proxy on the Wi-Fi AR performs the packet translation. Prior to the translation the proxy does a HIP base exchange with the default network gateway. The BEX consumes all available CPU cycles on Fonera for a period of five seconds (the first peak in Figure 8.4a), while on Avila the same operations result in less than 40% of the CPU utilization (the first peak in Figure 8.4b). After the BEX is completed, the HIP proxy on the Wi-Fi AR translates the plain TCP/IP packets it receives from the mobile client to the ESP packets it sends to the network gateway. As can be seen from the figures, the whole task is completed on Avila within 55 seconds. Fonera, on the contrary, due to limited resources spends on this operation more than 120 seconds.

Since the operations of packet translation require additional work (such as memory copying, database lookup, encryption operations, etc.), the throughput of the network is influenced by the amount of the available CPU cycles. We compared the TCP data throughput of the channel between the mobile client and the central network gateway with Avila running in the middle as a firewall and a proxy.

The results show that the HIP-aware firewall does not seriously affect the data rate, but the HIP proxy on Avila reduces the throughput by 8.7 Mbps (from 13.1 to 4.4 Mbps). In general, our measurements show that Avila offers a substantial increase in throughput in comparison to Fonera.

The memory usage becomes an issue when the HIP daemon is running in the proxy mode. In contrast to the firewall mode, 1 MB of available RAM on Fonera after the HIP daemon and proxy have been invoked is not sufficient. Due to absence of a swapping partition on the OpenWrt platform, the lack of RAM makes Fonera completely unresponsive with the only option of hard reset to bring it back. In contrast, the HIP proxy running on Avila with a notably larger amount of RAM can sustain several connections without problems. Though, when every packet is served in FIFO manner, per-packet processing time affects the overall system throughput.

8.4.3 Mode selection

The analysis of the measurement results allows us to give the following recommendations on deploying the architecture proposed in this article:

- For the areas with a small rate of connections, it is sufficient to have low cost devices such as La Fonera FON2100 to authenticate the users using HIT-based filtering on the distributed firewalls (i.e., running a HIP-aware firewall only).
- Since running only the HIP firewall does not require much of resources, one may consider using the existing Wi-Fi access routers but only modifying the software that is pre-installed on these devices.
- In cases when support for both plain unauthenticated and HIP authenticated traffic is needed (i.e., support for both legacy and HIP clients during the network transition state), more powerful devices such as Gateworks Avila GW2348-4 are required. However, even on powerful Wi-Fi ARs we recommend replacing a HIP proxy with another technique (e.g., forwarding packets to a demilitarized zone) to provide compatibility with legacy clients.

8.5 Summary

We have proposed a HIP-based distributed authentication architecture that can offer means to solve security and mobility issues in WLAN networks. The proposed

design is a two-level architecture where mobile users employ the Host Identity Protocol to connect to legacy Internet hosts through an operator's WLAN. The system includes an operator-specific proxy server and a distributed firewall running directly on WLAN ARs. The architecture has been implemented and validated on two different AR models with a Linux-based OpenWrt distribution.

Performance measurement results of HIP proxy and firewall running on the OpenWrt WLAN access routers have supported the motivation behind the two-level architecture. The hardware capabilities of the tested WLAN ARs are sufficient to run the HIP firewall performing a simple verification of the traffic based on the users' HITs. This prevents a malicious user from attacking the operator's internal infrastructure. Resource-intensive operations, such as serving as a HIP proxy and a target of the HIP base exchange, are given to a powerful server running in the fixed operator's network. The proxy enables a mobile user to benefit from the HIP properties such as IPsec encryption, mobility and multihoming, and IP cross-family support. The proposed architecture is planned to be deployed in a city-wide WLAN network in Northern Finland (panOULU).

9 Discussion

In this chapter we discuss the results of our empirical research. We provide our evaluation of feasibility of running the existing IP-based mobility and security mechanisms on lightweight hardware and make recommendations on the use of unmodified HIP on such devices. We conclude the chapter by discussing several prospective research directions.

9.1 HIP applicability to lightweight devices

In general, the obtained results indicate that the public-key cryptography and IPsec encryption involved with HIP are computationally expensive operations for lightweight mobile handhelds and can produce considerable delays to their users. Such operations can easily stress CPU, memory and battery resources of the devices such as the Nokia 770 and the Nokia E51 . However, this is a general statement and in practice the applicability of unmodified HIP to the resource-constrained mobile phones and PDAs should be considered depending on the QoS requirements of particular applications.

As an example, we conclude that unmodified HIP can be used in scenarios where a HIP-enabled mobile client communicates with remote Internet hosts through a single proxy server. In such a case, the establishment of a HIP association using the 1024-bit RSA keys takes 1.4 seconds on the average including two RTT of 2.5 ms for the Nokia 770 Internet Tablet. Since one HIP base exchange with the proxy is sufficient for the whole browsing session, most users will probably tolerate this delay. We make such conclusion based on the Nielsen's usability book [78], where the author elaborates on the "0.1/1/10" rule for the interactive applications, studied earlier by Miller [69] and Card *et al.* [11]. According to this rule, in 0.1 second the user should get a feedback showing that her action (e.g., a mouse click) was received; in 1 second the task should be completed to avoid the interruption of the user's work, otherwise an indicator with the task's completeness status should appear on the screen; finally, if in 10 seconds the task is not completed, the user loses her attention and most likely stops the task or switches to another one [78]. Based on this rule and assuming that each mobile application has an indicator (e.g., a status bar) for the completeness of a task, a 1.4 seconds-delay introduced by the HIP base exchange should not seriously decrease the user attention.

Nevertheless, the situation is likely to change when either two lightweight devices

communicate with each other through HIP or the client does not use proxy and, thus, needs to establish several associations with remote sites simultaneously. Our tests show that the time needed to set up a HIP association between two Nokia 770 Internet Tablets is above 2.6 seconds on the average and between two Nokia E51 smartphones varies from 3.5 to 6.4 seconds depending on the phone state (*active* or *standby*). The phone state is called *active* when its display is switched on and refreshing. In this state, a HIP base exchange duration representing a delay for the user is almost twice as long as in the *standby* state (3.2 versus 1.7 seconds). Because the user of a mobile phone most probably uses HIP with a GUI application that turns the phone into the *active* state, a HIP association establishment will always produce a three-second delay for the users of smartphones such as the Nokia E51.

The duration of a HIP mobility update on lightweight devices might be another concern if HIP is used with an application or a protocol that requires fast handovers. An update of a single HIP association between the Nokia 770 Internet Tablet and a remote peer takes 287 ms on the average with the RTT equal to 3.75 ms. Hence, the applications requiring the handoffs faster than this cannot efficiently utilize secure HIP mobility on such class of devices. Otherwise, this will likely cause poor performance and negative user experience. In case of a number of running applications and several open HIP associations, the mobile client will need to perform multiple update procedures upon changing its network attachment point, thus requiring even shorter delay per update. For comparison, a HIP mobility update between a 1.6-GHz IBM laptop and a server takes only 100 ms on the average with RTT equal to 1.6 ms. It is worth noting that these mobility measurements were performed 2,5 years ago with a code snapshot available at the time, so today's numbers might slightly change with the present implementations. In general, our experiments showed well a ratio between the handover durations on the lightweight Nokia 770 and a conventional IBM laptop.

One way to affect the application delay introduced by HIP is to adjust different parameters such as the length of a public key (RSA or DSA), the Diffie-Hellman key length and the puzzle difficulty. For instance, for applications that do not require strong security (e.g., web surfing) the duration of a HIP association establishment between a Nokia 770 and a server might be reduced to 0.4 seconds by using the 768-bit DH Group in the Diffie-Hellman key exchange and the 1024-bit RSA keys. Similarly, one might use a 512-bit RSA key instead of a 1024-bit key. However, reducing the key lengths is rather unfavourable solution because it would result in a less secure communication context increasing the probability of attacks. Contrary, one might want to increase the length of the RSA and DH keys in an untrustworthy

environment. This in turn will cause even longer delays to exchange four HIP BEX and three mobility update packets.

Our measurements with different values of the puzzle difficulty showed that reducing the default value of $K = 10$ does not significantly decrease processing time of such a puzzle by the *Initiator* whereas raising it to $K = 15$ and above increases the processing time exponentially on the Nokia 770 Internet Tablet. As an enhancement for parameter adjustments, it would be useful to have a mechanism that can automatically detect the type of the client hardware prior to communications and depending on the available resources offer to lightweight peers smaller key lengths and puzzle difficulty value.

Taking into consideration the CPU utilization perspective our results with a Symbian-based Nokia E51 indicate a 100% processor load during the intensive HIP daemon operations such as initial generation of a public-private key pair and establishment of a HIP association. Otherwise, after the base exchange is completed and the HIP daemon falls into the standby mode, the CPU utilization is minimal. The memory usage on the Nokia E51 in our experiments proved to be within normal bounds during the HIP base exchange raising the total amount of used RAM by only 3-4 MB on the average.

In another set of the experiments we evaluated the impact of the IPsec ESP data encryption involved with HIP on RTT and TCP throughput. In general, the RTT with ESP was slightly higher than over plain IP, except for the first RTT that triggered a HIP base exchange (that value was over 1 second and we did not include it in the calculated average). With the initial RTT=2.08 ms, the ESP encryption increased this value by just 0.67 ms. It would be unfair to generalize these results because the RTT differs in different networks.

The maximum achievable TCP throughput in our experiments with the Nokia 770 over a wireless link was initially constrained either by improper driver implementation, low CPU power or another reason. An average value of 4.86 Mbps appeared to be an upper bound for the Internet Tablet in an encryption-free network. The use of HIP and ESP further reduced the throughput measured with *iperf* by 1.59 Mbps. We also compared ESP with WPA encryption enabled on the wireless access router. Implemented on hardware WPA expectedly produced tiny impact as compared to software-based IPsec ESP. The comparison with a 1.6-GHz IBM laptop showed that HIP affects more seriously devices with low computational power. The experiments with the Nokia 770 and the laptop were performed in equal or similar conditions. Hence we assume that the difference in numbers obtained on both devices shows

us the pure effect of IPsec ESP encryption. However, we did not control all factors that could potentially affect the measurements results. For instance, we assumed that the WLAN signal strength was merely equal on both Nokia tablet and IBM laptop connected to the same Wi-Fi AR and placed in the same proximity from it. Yet we did not measure the actual signal strength and cannot provide any numbers. Similarly, we did not control different TCP options that can affect the throughput. Ideally, it is necessary to distinguish the effects on the throughput made by the application, by TCP itself and by the network channel [94]. Due to its complexity, we leave such analysis out of the thesis's scope.

Our preliminary results on power consumption indicate that the use of ESP encryption with HIP does not notably affect the current consumption on the Nokia 770, although the energy cost per byte is higher with HIP due to reduced throughput. We noticed that the Tablet's CPU is always fully utilized when an application transmits data over WLAN, and this depletes the battery in 3-4 hours. The measurements of power consumption and evaluation of the HIP effect on the battery lifetime on Nokia E51 showed 222 mW/60 mA and 333 mW/90 mA for the cases when the phone was in the "normal" use (i.e., no HIP daemon was running) and with the HIP daemon doing a base exchange. The observed values correspond to 18 and 12 hours of a 1050-mAh battery. In reality, the constant exchange of the HIP control packets is abnormal behaviour of the HIP daemon, so the purpose of these results is rather to illustrate an instant power consumption by the cryptography operations constituting the BEX.

9.2 Future research directions

The existing mobile phones, PDAs, Wi-Fi routers, remote controllers, sensors and many other embedded appliances in our daily life are increasingly utilizing the TCP/IP communication stack to interconnect between each other. This tendency has received its own definition - the *"Internet of Things"* [42], which considers any small object as a potential participant of a physical IP network. Originating from the early 2000s, the idea of the "Internet of Things" heavily relies on two important domains: how to identify objects and detect their changes, and how to connect them. The first domain is to a large extent based on the RFID and sensor technologies, whereas the second is attributed to the TCP/IP communication stack.

In this thesis, we addressed selected important security and mobility issues of the *mobile Internet*. In particular, we empirically evaluated the applicability of the Host Identity Protocol to three classes of lightweight devices including a Linux-based

PDA, a Symbian-based smartphone and two models of the OpenWrt-based Wi-Fi access routers. Since these devices are a definite part of the future all-IP networks, we classify our present work to fall into the communication domain of the "Internet of Things". As a part of the future research, we consider important to further investigate this emerging concept by focusing on its two domains: identification and access control for the embedded objects and secure communications between them. As an example, the applicability of existing IP security mechanisms to the different types of sensor networks can be evaluated.

Besides communications security in a network composed of sensors or other lightweight objects, one important perspective for our future considerations is energy consumption. While in this work we focused on measuring the effect of public-key cryptography and IPsec encryption on the battery lifetime of different handhelds, another factor strongly influencing power consumption of such devices is data transmission over a wireless interface. Ideally, a wireless smart object should stay most of the time in a sleep mode and wake up for only short periods to exchange data with its peers. A variety of approaches has been proposed to eliminate idle states of lightweight objects in wireless ad-hoc networks and thus achieve rational utilization of their energy resources. A substantial contribution in this area, potentially valuable to our future research, belongs to Feeney *et al.*. In their works [20, 21, 22] the authors studied multiple power save protocols in wireless networks and evaluated the impact of different node wakeup algorithms on network capacity and energy efficiency.

One interesting recent initiative in the field of the "Internet of Things" is the industry alliance *IPSO* (IP for Smart Objects) [18]. IPSO's mission is to encourage different organisations and parties in the use of the Internet Protocol as the most standardized and non-proprietary solution for networking of small objects. IPSO's activities do not substitute but complement the work performed by the IETF and IEEE forums by documenting new IP-based technologies intended for embedded objects, making interoperability tests between different smart devices, applications and services, and serving as an informative, promotion and marketing entity [18].

As a potential continuation of our present research and a target for future work, we consider the (wireless) "Internet of Things" as an environment where different embedded devices (from a light bulb to a temperature meter) coexist, are interconnected and possibly controlled remotely (e.g., by mobile terminals). In such an "ecosystem", we must ensure secure and efficient mobile communications between all lightweight participants. In achieving this goal, we will most likely need to alleviate weaknesses identified in this thesis, e.g., the computational expenses of the

public-key cryptography and IPsec encryption. The protocols operating in such networks should be adaptable to different classes of communicating devices. Further comparison of our empirical results against related work will provide more insights into the research target and assist in finding alternative approaches for security on lightweight mobile devices.

10 Conclusions

In this thesis, we addressed the aspect of secure communications in the mobile Internet. In particular, we considered selected security and mobility issues on lightweight mobile devices and network components. Our main objectives were to evaluate the feasibility of running the existing IP-layer security and mobility protocols (such as HIP and IPsec) on lightweight mobile devices and assess the impact of constrained hardware resources on performance of the protocols' operations.

We performed a literature study to obtain the insights into the research target and evaluated the contribution of the related work. In the empirical part of our research we conducted a number of experiments involving several mobile clients with limited CPU, memory and battery resources running the Host Identity Protocol. We measured a set of different HIP and network-related performance metrics. The analysis of the results allowed us to make recommendations on using unmodified public-key cryptography mechanisms on lightweight devices in the mobile Internet. In addition, we reported our development and porting experience, which can be useful for migration of OSS projects to an embedded platform.

Our empirical results indicate that unmodified HIP and IPsec are feasible to run on mobile devices and lightweight network components without a significant performance degradation only in particular cases. That is why the evaluation of HIP and similar protocols has to distinguish between different scenarios and applications, taking into account varying QoS requirements.

Our experimentations with two different Wi-Fi access router models showed that the choice of the AR hardware is dictated by the requirements to the serving capacity (i.e., the amount of clients) and the underlying network architecture (i.e., which functions are performed by the Wi-Fi ARs).

One of the important future research direction is to study different IP security protocols and their adaptability for distinct classes of communicating devices in the mobile Internet, which can be achieved, e.g., by the use of flexible protocol parameters. Another approach can be to concentrate on secure communications in the "Internet of Things" and, in addition to mobile handhelds, consider other smart objects.

References

[1] HIPL website. [Online] Available at: `http://infrahip.hiit.fi` Accessed 23 April 2009.

[2] OpenHIP website. [Online] Available at: `http://www.openhip.org` Accessed 23 April 2009.

[3] panOULU network website. [Online] Available at: `http://www.panoulu.net`. Accessed 23 April 2009.

[4] Symbian Fast Facts Q2 2008. [Online] Available at: `http://www.symbian.com/about/fast.asp`. Accessed 23 April 2009.

[5] J. Abeillé, M. Durvy, J. Hui, and S. Dawson-Haggerty. 2008. Lightweight IPv6 Stacks for Smart Objects: the Experience of Three Independent and Interoperable Implementations. Internet Protocol for Smart Objects (IPSO) Alliance. White Paper #2. [Online] Available at `http://www.ipso-alliance.org/Documents/IPSO-WP-2.pdf`. Accessed 20 May 2009.

[6] J. Arkko, C. Vogt, and W. Haddad. 2007. Enhanced Route Optimization for Mobile IPv6. RFC 4866, IETF.

[7] F. Baboescu and G. Varghese. 2005. Scalable packet classification. IEEE/ACM Trans. Netw. 13, no. 1, pages 2–14.

[8] K. Brasee, S. K. Makki, and S. Zeadally. 2008. A Novel Distributed Authentication Framework for Single Sign-On Services. In: SUTC '08: Proc. of the 2008 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing, pages 52–58. IEEE Computer Society, Washington, DC, USA. ISBN 978-0-7695-3158-8.

[9] G. Camarillo, I. Mas, and P. Nikander. 2008. A Framework to Combine the Session Initiation Protocol and the Host Identity Protocol. In: Proc. of Wireless Communications and Networking Conference, 2008 (WCNC 2008). IEEE, pages 3051–3056.

[10] A. Campbell, J. Gomez, S. Kim, and C.-Y. Wan. 2002. Comparison of IP Micromobility Protocols. IEEE Wireless Communications .

[11] S. K. Card, G. G. Robertson, and J. D. Mackinlay. 1991. The information visualizer, an information workspace. In: CHI '91: Proceedings of the SIGCHI conference on Human factors in computing systems, pages 181–186. ACM, New York, NY, USA. ISBN 0-89791-383-3.

[12] Certicom Research. 2000. Standards for Efficient Cryptography. SEC 2: Recommended Elliptic Curve Domain Parameters. Version 1.0, Certicom Research.

[13] Certicom Research. 2008. Standards for Efficient Cryptography. SEC 1: Elliptic Curve Cryptography. Working draft. version 1.9, Certicom Research.

[14] J.-S. Coron. 2006. What is cryptography? Security & Privacy, IEEE 4, no. 1, pages 70–73.

[15] V. Devarapalli, R. Wakikawa, A. Petrescu, and P. Thubert. 2005. Network Mobility (NEMO) Basic Support Protocol. RFC 3963, IETF.

[16] W. Diffie and M. E. Hellman. 1976. New Directions in Cryptography. IEEE Transactions on Information Theory IT-22, no. 6, pages 644–654.

[17] A. Dunkels. 2003. Full TCP/IP for 8-bit architectures. In: MobiSys '03: Proceedings of the 1st International Conference on Mobile Systems, Applications and Services, pages 85–98. ACM, New York, NY, USA.

[18] A. Dunkels and J.-P. Vasseur. 2008. IP for Smart Objects. Internet Protocol for Smart Objects (IPSO) Alliance. White Paper #1. [Online] Available at `www.ipso-alliance.org/Documents/IPSO-WP-1.pdf`. Accessed 18 May 2009.

[19] M. Durvy, J. Abeillé, P. Wetterwald, C. O'Flynn, B. Leverett, E. Gnoske, M. Vidales, G. Mulligan, N. Tsiftes, N. Finne, and A. Dunkels. 2008. Making Sensor Networks IPv6 Ready. In: Proc. of the Sixth ACM Conference on Networked Embedded Sensor Systems (ACM SenSys 2008), poster session. Raleigh, North Carolina, USA.

[20] L. M. Feeney, B. Ahlgren, and P. Gunningberg. 2005. Enabling adaptive traffic scheduling in asynchronous miltihop wireless networks. In: Proc. of the Third Swedish National Computer Networking Workshop (SNCNW 2005), page 4. Halmstad, Sweden. `http://eprints.sics.se/262/01/snowcow05enabling.pdf`.

[21] L. M. Feeney, C. Rohner, and B. Ahlgren. 2007. The impact of wakeup schedule distribution in synchronous power save protocols on the performance of multihop wireless networks. In: Proc. of the IEEE Wireless Communications and Networking Conference (WCNC'07). Hong Kong. http://eprints.sics.se/2631/01/wcnc07impact.pdf.

[22] L. M. Feeney, C. Rohner, and B. Ahlgren. 2007. Leveraging a power save protocol to improve performance in ad hoc networks. SIGMOBILE Mob. Comput. Commun. Rev. 11, no. 2, pages 51–52.

[23] D. Forsberg. 2007. Secure Distributed AAA with Domain and User Reputation. In: Proc. of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007, pages 1–6. IEEE Computer Society.

[24] Forum Nokia. 2005. Symbian OS: Overview To Networking. [Online] Available at: http://sw.nokia.com/id/c4536832-3dd0-45af-94be-1c4289cc3003/Symbian_OS_Overview_To_Networking_v1_0_en.pdf. Accessed 18 May 2009.

[25] Forum Nokia. 2009. Nokia Energy Profiler Quick Start Guide. [Online] Available: http://www.forum.nokia.com/Resources_and_Information/Explore/Development_Process_and_User_Experience/Power_Management/Nokia_Energy_Profiler_Quick_Start.xhtml. Accessed 18 May 2009.

[26] Forum Nokia. 2009. Open C API Reference. [Online] Available: http://library.forum.nokia.com/index.jsp?topic=/S60_5th_Edition_Cpp_Developers_Library/GUID-FE27AB35-C6FD-4F11-802D-0D5FCFFC2976/html/mrt/Open_C_API_ReferenceIndexPage.html. Accessed 18 May 2009.

[27] Forum Nokia. 2009. Open C/C++ Documentation. [Online] Available at: http://www.forum.nokia.com/Resources_and_Information/Documentation/Open_C_and_C++.xhtml. Accessed 18 May 2009.

[28] R. Good and N. Ventura. 2006. A Multilayered Hybrid Architecture to Support Vertical Handover Between IEEE802.11 and UMTS. In: IWCMC '06: Proceedings of the 2006 international conference on Wireless communications and mobile computing, pages 257–262. ACM, New York, NY, USA. ISBN 1-59593-306-9.

[29] A. Gurtov. 2008. Host Identity Protocol (HIP): Towards the Secure Mobile Internet. Wiley and Sons. ISBN 978-0-470-99790-1.

[30] H. Hamed and E. Al-Shaer. 2006. Dynamic rule-ordering optimization for high-speed firewall filtering. In: ASIACCS '06: Proceedings of the 2006 ACM Symposium on Information, computer and communications security, pages 332–342. ACM, New York, NY, USA. ISBN 1-59593-272-0.

[31] M. Haque, A.-S. Pathan, and C. S. Hong. 2008. Securing U-Healthcare Sensor Networks using Public Key Based Scheme. In: Proc. of the 10th International Conference on Advanced Communication Technology, 2008 (ICACT 2008), volume 2, pages 1108–1111.

[32] L.-S. He and N. Zhang. 2003. An Asymmetric Authentication Protocol for M-Commerce Applications. In: ISCC '03: Proceedings of the Eighth IEEE International Symposium on Computers and Communications, page 244. IEEE Computer Society, Washington, DC, USA. ISBN 0-7695-1961-X.

[33] T. Heer. 2006. LHIP: Lightweight Authentication for the Host Identity Protocol (HIP). Master's thesis, University of Tubingen, Protocol-Engineering&Distributed Systems research group.

[34] T. Heer. 2007. LHIP Lightweight Authentication Extension for HIP: draft-heer-hip-lhip-00.txt. Work in progress.

[35] T. Heer, S. Götz, E. Weingärtner, and K. Wehrle. 2008. Secure Wi-Fi Sharing at Global Scales. In: Proc. of the 15th International Conference on Telecommunications (ICT 2008). IEEE, St. Petersburg, Russian Federation.

[36] T. Heer, R. Hummen, M. Komu, S. Götz, and K. Wehrle. 2009. End-host Authentication and Authorization for Middleboxes based on a Cryptographic Namespace. In: Proc. of the IEEE International Conference on Communications 2009 (ICC 2009). IEEE, Dresden, Germany. To appear.

[37] T. Heer, S. Li, and K. Wehrle. 2007. PISA: P2P Wi-Fi Internet Sharing Architecture. In: Proc. of the 7th International Conference on Peer-to-Peer Computing. Galway, Ireland.

[38] T. R. Henderson. 2003. Host Mobility for IP Networks: A Comparison. IEEE Network 17, no. 6, pages 18–26.

[39] T. R. Henderson. 2004. CAN SIP USE HIP. In: HIP Workshop, 61st IETF meeting.

[40] T. R. Henderson, J. M. Ahrenholz, and J. H. Kim. 2003. Experience with the Host Identity Protocol for Secure Host Mobility and Multihoming. In: Proc. of the IEEE Wireless Communications and Networking Conference (WCNC'03).

[41] S. Herborn, L. Haslett, R. Boreli, and A. Seneviratne. 2006. HarMoNy - HIP Mobile Networks. In: Proc. of the IEEE 63rd Vehicular Technology Conference (VTC '06), pages 871–875.

[42] International Telecommunication Union. 2005. ITU Internet Reports 2005: The Internet of Things. Executive Summary. [Online] Available at: `http://www.itu.int/dms_pub/itu-s/opb/pol/S-POL-IR.IT-2005-SUM-PDF-E.pdf`. Accessed 18 May 2009.

[43] S. Ioannidis, A. D. Keromytis, S. M. Bellovin, and J. M. Smith. 2000. Implementing a Distributed Firewall. In: CCS '00: Proc. of the 7th ACM Conference on Computer and Communications Security, pages 190–199. ACM, New York, NY, USA. ISBN 1-58113-203-4.

[44] C. Jian, R. Yan, Z. Hongke, and Z. Sidong. 2005. A proposal to replace HIP base exchange with IKE-H method: draft-yan-hip-ike-h-02. Work in progress. Expired in May, 2006.

[45] W. Jiang and V. K. Prasanna. 2009. Large-scale wire-speed packet classification on FPGAs. In: FPGA '09: Proceeding of the ACM/SIGDA international symposium on Field programmable gate arrays, pages 219–228. ACM, New York, NY, USA. ISBN 978-1-60558-410-2.

[46] D. B. Johnson, C. Perkins, and J. Arkko. 2004. Mobility Support in IPv6. RFC 3775, IETF.

[47] P. Jokela, R. Moskowitz, and P. Nikander. 2008. Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP). IETF RFC 5202. URL `http://tools.ietf.org/html/rfc5202`.

[48] P. Jokela, T. Rinta-Aho, T. Jokikyyny, J. Wall, M. Kuparinen, H. Mahkonen, J. Melen, T. Kauppinen, and J. Korhonen. 2004. Handover Performance with HIP and MIPv6. In: Proc. of the 1st International Symposium on Wireless Communication Systems, ISWCS'04.

[49] J.-W. Jung, H.-K. Kahng, R. Mudumbai, and D. Montgomery. 2003. Performance Evaluation of Two Layered Mobility Management Using Mobile IP and Session Initiation Protocol. In: Proc. of Global Telecommunications Conference. GLOBECOM '03, volume 3, pages 1190–1194.

[50] C. Kaufman. 2005. Internet Key Exchange (IKEv2) Protocol. RFC 4306, IETF.

[51] S. Kent and R. Atkinson. 1998. IP Authentication Header. RFC 2402 (Proposed Standard). URL `http://www.ietf.org/rfc/rfc2402.txt`. Obsoleted by RFC 4302.

[52] S. Kent and R. Atkinson. 1998. IP Encapsulating Security Payload (ESP). RFC 2406, IETF.

[53] A. Khurri, D. Kuptsov, and A. Gurtov. 2009. Performance of Host Identity Protocol on Symbian OS. In: Proc. of the IEEE International Conference on Communications 2009 (ICC'09). IEEE.

[54] A. Khurri, E. Vorobyeva, and A. Gurtov. 2007. Performance of Host Identity Protocol on Lightweight Hardware. In: Proc. of the 2nd ACM/IEEE International Workshop on Mobility in the Evolving Internet Architecture (MobiArch'07). ACM, New York, NY, USA. ISBN 978-1-59593-784-8.

[55] N. Koblitz. 1987. Elliptic Curve Cryptosystems. Mathematics of Computation, 1987 48, no. 177, pages 203–209.

[56] N. Koblitz, A. Menezes, and S. Vanstone. 2000. The State of Elliptic Curve Cryptography. Designs, Codes and Cryptography 19, no. 2-3, pages 173–193.

[57] T. Koponen, A. Gurtov, and P. Nikander. 2005. Application mobility with Host Identity Protocol. In: Proc. of NDSS Wireless and Security Workshop. Internet Society, San Diego, CA, USA.

[58] J. Korhonen, A. Mäkela, and T. Rinta-aho. 2007. HIP Based Network Access Protocol in Operator Network Deployments. In: Proc. of the First Ambient Networks Workshop on Mobility, Multiaccess, and Network Management (M2NM'07).

[59] J. Korhonen. 2008. IP Mobility in Wireless Operator Networks. Ph.D. thesis, University of Helsinki, Department of Computer Science, P.O. Box 68, FIN-00014 University of Helsinki, Finland.

[60] D. Kuptsov and A. Gurtov. 2009. SAVAH: Source Address Validation with Host Identity Protocol. In: Proc. of the First International ICST Conference on Security and Privacy in Mobile Information and Communication Systems (MobiSec'09).

[61] D. Kuptsov, A. Khurri, and A. Gurtov. 2009. Distributed User Authentication in Wireless LANs. In: Proc. of the 10th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'09). IEEE.

[62] J. Laganier and L. Eggert. 2008. Host Identity Protocol (HIP) Rendezvous Extension. IETF RFC 5204. URL `http://tools.ietf.org/html/rfc5204`.

[63] J. Laganier, T. Koponen, and L. Eggert. 2008. Host Identity Protocol (HIP) Registration Extension. IETF RFC 5203. URL `http://tools.ietf.org/html/rfc5203`.

[64] H. Lee, S. W. Lee, and D. Cho. 2003. Mobility management based on the integration of mobile IP and session initiation protocol in next generation mobile data networks. In: Proceedings of IEEE 58th Vehicular Technology Conference VTC 2003Fall, volume 3, pages 2058–2062.

[65] J. Lim, M. Han, and J. Kim. 2005. Implementation of light-weight IKE protocol for IPsec VPN within router. In: Proc. of the 7th International Conference on Advanced Communication Technology (ICACT 2005), volume 1, pages 81–84.

[66] K. Malhotra, S. Gardner, and R. Patz. 2007. Implementation of Elliptic-Curve Cryptography on Mobile Healthcare Devices. In: Proc. of the IEEE International Conference on Networking, Sensing and Control, 2007, pages 239–244.

[67] J. Manner and M. Kojo. 2004. Mobility Related Terminology. RFC 3753, IETF.

[68] D. L. McDonald, C. W. Metz, and B. G. Phan. 2005. PF_KEY Key Management API, Version 2: draft-mcdonald-pf-key-v2-05. Work in progress.

[69] R. B. Miller. 1968. Response time in man-computer conversational transactions. In: AFIPS '68 (Fall, part I): Proceedings of the December 9-11, 1968, fall joint computer conference, part I, pages 267–277. ACM, New York, NY, USA.

[70] V. S. Miller. 1986. Use of Elliptic Curves in Cryptography. In: Lecture notes in computer sciences; 218 on Advances in cryptology—CRYPTO 85, pages 417–426. Springer-Verlag New York, Inc., New York, NY, USA. ISBN 0-387-16463-4.

[71] R. Moskowitz and P. Nikander. 2006. Host Identity Protocol Architecture. IETF RFC 4423. URL http://www.ietf.org/rfc/rfc4423.txt.

[72] R. Moskowitz, P. Nikander, P. Jokela, and T. R. Henderson. 2007. Host Identity Protocol: draft-ietf-hip-base-10. Work in progress. Expires in May, 2008.

[73] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson. 2008. Experimental Host Identity Protocol (HIP). IETF RFC 5201.

[74] National Institute of Standards and Technology. 1999. FIPS PUB 46-3: Data Encryption Standard (DES). pub-NIST. [Online]. Available at http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf. Accessed 18 May 2009.

[75] National Institute of Standards and Technology. 2000. FIPS PUB 186-2: Digital Signature Standard (DSS). pub-NIST, pub-NIST:adr. [Online]. Available at http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf. Accessed 18 May 2009.

[76] National Institute of Standards and Technology. 2001. FIPS PUB 197, Advanced Encryption Standard (AES). pub-NIST. [Online]. Available at http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf. Accessed 18 May 2009.

[77] National Security Agency. 2009. The Case for Elliptic Curve Cryptography. [Online]. Available at: http://www.nsa.gov/business/programs/elliptic_curve.shtml. Accessed 18 May 2009.

[78] J. Nielsen. 1993. Usability Engineering. AP Professional. ISBN 0-12-518405-0.

[79] P. Nikander, T. Henderson, C. Vogt, and J. Arkko. 2008. End-Host Mobility and Multihoming with the Host Identity Protocol (HIP). IETF RFC 5206. URL http://tools.ietf.org/html/rfc5206.

[80] P. Nikander and J. Laganier. 2008. Host Identity Protocol (HIP) Domain Name System (DNS) Extension. IETF RFC 5205. URL http://tools.ietf.org/html/rfc5205.

[81] P. Nikander and J. Melen. 2008. A Bound End-to-End Tunnel (BEET) mode for ESP: draft-nikander-esp-beet-mode-09. URL http://tools.ietf.org/html/draft-nikander-esp-beet-mode-09. Work in progress.

[82] P. Nikander, J. Ylitalo, and J. Wall. 2003. Integrating Security, Mobility, and Multi-Homing in a HIP Way. In: Proc. of Network and Distributed Systems Security Symposium (NDSS'03). Internet Society, San Diego, CA, USA. ISBN 1-891562-16-9.

[83] E. Nordmark and M. Bagnulo. 2009. Shim6: Level 3 Multihoming Shim Protocol for IPv6. Internet-Draft (work in progress) 12, IETF. URL http://tools.ietf.org/html/draft-ietf-shim6-proto-12.

[84] S. Novaczki, L. Bokor, and S. Imre. 2006. Micromobility Support in HIP: Survey and Extension of Host Identity Protocol. In: Proc. of the IEEE Mediterranean Electrotechnical Conference (MELECON 2006), pages 651–654.

[85] S. Novaczki, L. Bokor, and S. Imre. 2007. A HIP Based Network Mobility Protocol. In: Proc. of the International Symposium on Applications and the Internet Workshops, 2007. SAINT Workshops 2007, pages 48–48.

[86] P. Eronen. 2006. IKEv2 Mobility and Multihoming Protocol (MOBIKE). RFC 4555 (Proposed Standard). URL http://tools.ietf.org/html/rfc4555.

[87] P. Pääkkönen, P. Salmela, R. Aguero, and J. Choque. 2008. Performance Analysis of HIP-based Mobility and Triggering. In: Proc. of the International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'08).

[88] C. Perkins. 2002. IP Mobility Support for IPv4. RFC 3334, IETF. URL http://tools.ietf.org/html/rfc3344.

[89] R. L. Rivest, A. Shamir, and L. M. Adelman. 1977. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Technical Report MIT/LCS/TM-82. URL citeseer.ist.psu.edu/rivest78method.html.

[90] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, and E. Schooler. 2002. SIP: Session Initiation Protocol. RFC 3261, IETF.

[91] B. Schneier. 1993. Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish). Fast Software Encryption, Cambridge Security Workshop Proceedings pages 191–204.

[92] H. Schulzrinne and E. Wedlund. 2000. Application-layer mobility using SIP. SIGMOBILE Mobile Computing and Communications Review 4, no. 3, pages 47–57.

[93] C. Shannon. 1949. Communication Theory and Secrecy Systems. Bell Telephone Laboratories.

[94] M. Siekkinen, G. Urvoy-Keller, E. W. Biersack, and D. Collange. 2008. A root cause analysis toolkit for TCP. Comput. Netw. 52, no. 9, pages 1846–1858.

[95] J. Y. H. So, J. Wang, and D. Jones. 2005. SHIP - Mobility Management Hybrid SIP-HIP Scheme. In: Proc. of the 6th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Networks (SNPD/SAWN'05), pages 226–230.

[96] J. So and J. Wang. 2008. Micro-HIP A HIP-Based Micro-Mobility Solution. In: Proc. of the IEEE International Conference on Communications Workshops, 2008. ICC Workshops'08, pages 430–435.

[97] H. Soliman, C. Catelluccia, K. E. Malki, and L. Bellier. 2004. Hierarchical Mobile IPv6 mobility management (HMIPv6): draft-ietf-mipshop-hmipv6-04. Internet draft, IETF. Work in progress. Expires in June 2005.

[98] R. Stepanek. 2001. Distributed Firewalls. In: Publications in Telecommunication Software and Multimedia. Helsinki University of Technology. [Online]. Available at `http://www.tml.tkk.fi/Studies/T-110.501/2001/papers/robert.stepanek.pdf`. Accessed 18 May 2009.

[99] L. Thames, R. Abler, and D. Keeling. 2009. Bit vector algorithms enabling high-speed and memory-efficient firewall blacklisting. In: ACM-SE 47: Proceedings of the 47th Annual Southeast Regional Conference, pages 1–6. ACM, New York, NY, USA. ISBN 978-1-60558-421-8.

[100] H. Wang, B. Sheng, C. C. Tan, and Q. Li. 2008. Comparing Symmetric-key and Public-key Based Security Schemes in Sensor Networks: A Case Study of User Access Control. In: ICDCS '08: Proceedings of the 2008 The 28th International Conference on Distributed Computing Systems, pages 11–18. IEEE Computer Society, Washington, DC, USA. ISBN 978-0-7695-3172-4.

[101] Q. Wang and M. Abu-Rgheff. 2006. Mobility management architectures based on joint mobile IP and SIP protocols. IEEE Wireless Communications 13, no. 6, pages 68–76.

[102] K. D. Wong, A. Dutta, J. Burns, R. Jain, and K. Young. 2003. A Multilayered Mobility Management Scheme for Auto-Configured Wireless IP Networks. IEEE Wireless Communications 10, no. 5, pages 62–69.

[103] J. Wu, G. Ren, J. Bi, X. Li, R. Bonica, and M. Williams. 2007. Source Address Validation Architecture (SAVA) Framework: draft-wu-sava-framework-00.txt. Work in progress.

[104] J. Wu, G. Ren, J. Bi, X. Li, and M. Williams. 2007. A First-Hop Source Address Validation Solution for SAVA: draft-wu-sava-solution-firsthop-eap-00. Work in progress.

[105] J. Wu, G. Ren, and X. Li. 2007. Source Address Validation: Architecture and Protocol Design. In: Proc. of the IEEE International Conference on Network Protocols, pages 276–283.

[106] J. Ylitalo. 2005. Re-thinking Security in Network Mobility. In: Proc. of NDSS Wireless and Security Workshop. Internet Society, San Diego, CA, USA.

[107] J. Ylitalo. 2008. Secure Mobility at Multiple Granularity Levels over Heterogeneous Datacom Networks. Ph.D. thesis, Helsinki University of Technology, Department of Computer Science and Engineering, P.O. Box 5400, FI-02015 TKK, Finland.

[108] J. Ylitalo, J. Melen, P. Nikander, and V. Torvinen. 2004. Re-thinking Security in IP based Micro-Mobility. In: Proc. of the 7th Information Security Conference (ISC04).