Department of Computer Science and Engineering

Evaluating IP Security on Lightweight Hardware

Andrey Khurri

Aalto University

DOCTORAL DISSERTATIONS

Aalto University publication series
DOCTORAL DISSERTATIONS 2/2011

Evaluating IP Security on Lightweight Hardware

Andrey Khurri

Doctoral dissertation for the degree of Doctor of Science in Technology to be presented with due permission of the Faculty of Information and Natural Sciences for public examination and debate in Auditorium T2 at the Aalto University School of Science (Espoo, Finland) on the 24th of January 2011 at 12 noon.

Aalto University
School of Science
Department of Computer Science and Engineering

Supervisor
Professor Antti Ylä-Jääski

Instructor
Professor Dr. Andrei Gurtov

Preliminary examiners
Professor Dr.-Ing. Klaus Wehrle
Dr. Bengt Ahlgren

Opponent
Dr. Kevin Fall

Aalto University publication series
DOCTORAL DISSERTATIONS 2/2011

ISBN 978-952-60-4004-2 (print)
ISBN 978-952-60-4005-9 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)

Aalto Print
Helsinki 2011

The dissertation can be read at http://lib.tkk.fi/Diss/

Publications orders (printed book):
www.aalto.fi
School of Science
Department of Computer Science and Engineering

Aalto University

Abstract

Aalto University, P.O. Box 11000, FI-00076 Aalto www.aalto.fi

Author
Andrey Khurri

Name of the doctoral dissertation
Evaluating IP Security on Lightweight Hardware

Publisher Aalto University School of Science

Unit Department of Computer Science and Engineering

Series Aalto University publication series DOCTORAL DISSERTATIONS 2/2011

Field of research Data Communications Security

Manuscript submitted 24.08.2010 **Manuscript revised** 07.01.2011

Date of the defence 24.01.2011 **Language** English

☒ **Monograph** ☐ **Article dissertation (summary + original articles)**

Abstract

TCP/IP communications stack is being increasingly used to interconnect mobile phones, PDAs, sensor motes and other wireless embedded devices. Although the core functionality of communications protocols has been successfully adopted to lightweight hardware from the traditional Internet and desktop computers, suitability of strong security mechanisms on such devices remains questionable. Insufficient processor, memory and battery resources, as well as constraints of wireless communications limit the applicability of many existing security protocols that involve computationally intensive operations. Varying capabilities of devices and application scenarios with different security and operational requirements complicate the situation further and call for agile and flexible security systems.

This study does an empirical evaluation of applicability of selected existing IP security mechanisms to lightweight (resource-constrained) devices. In particular, we evaluate various components of the Host Identity Protocol (HIP), standardized by the Internet Engineering Task Force for achieving authentication, shared key negotiation, secure mobility and multihoming and, if used with IPsec, integrity and confidentiality of user data. Involving a set of cryptographic operations, HIP might easily stress a lightweight client, while affecting performance of applications running on it and shortening battery lifetime of the device.

We present a background and related work on network-layer security, as well as a set of measurement results of various security components obtained on devices representing lightweight hardware: embedded Linux PDAs, Symbian-based smartphones, OpenWrt Wi-Fi access routers and wireless sensor platforms. To improve computational and energy efficiency of HIP, we evaluate several lightweight mechanisms that can substitute standard protocol components and provide a good trade-off between security and performance in particular application scenarios. We describe cases where existing HIP security mechanisms (i) can be used unmodified and (ii) should be tailored or replaced to suit resource-constrained environments. The combination of presented security components and empirical results on their applicability can serve as a reference framework for building adaptable and flexible security services for future lightweight communication systems.

Keywords Host Identity Protocol, IP security, cryptography, performance, resource-constrained devices, mobile Internet

ISBN (printed) 978-952-60-4004-2 **ISBN (pdf)** 978-952-60-4005-9

ISSN-L 1799-4934 **ISSN (printed)** 1799-4934 **ISSN (pdf)** 1799-4942

Pages 150 **Location of publisher** Espoo **Location of printing** Helsinki **Year** 2011

The dissertation can be read at http://lib.tkk.fi/Diss/

To my grandparents

Acknowledgements

This thesis is the result of my four-year work at Helsinki Institute for Information Technology and postgraduate studies at Helsinki University of Technology (Aalto University School of Science and Technology from the beginning of 2010). The work presents accumulated research on the Host Identity Protocol and security components involved with it applied to resource-constrained mobile phones and PDAs, wireless routers and wireless sensor devices.

The results and findings presented in this thesis are to a large extent an outcome of our joint work with many people in HIIT, Aalto and partner organizations. Without your support it would have been unrealistic to achieve my goals. I would like to express my deepest gratitude to my supervisor Prof. Dr. Tech. Antti Ylä-Jääski and my instructor 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 for supervising this thesis and providing useful advice, comments and specific guidelines concerning my postgraduate studies. Thank you, Antti, for being always supportive, for your assistance and positive attitude. I wish to thank also the Faculty of Information and Natural Sciences for funding the last phase of my doctoral studies.

The research 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, René Hummen and Stefan Götz.

I would like to thank Prof. Klaus Wehrle and Dr. Bengt Ahlgren for acting as the pre-examiners of the dissertation, for the time spent on evaluating my work and for providing valuable comments and feedback.

I am indebted to 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, Dmitriy Kuptsov, Andrey Lukyanenko, Boris Nechaev, Kristiina Karvonen, Joakim Koskela, Oleg Ponomarev, Samu Varjonen, Juho Heikkilä, Theofanis Kilinkaridis and Tatiana Polishchuk, my sincere thanks go to all of you. I would like to thank the rest of my colleagues from HIIT, 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, Andrea, Lotta, Marja-Leena and Visa for your continuous assistance with all administrative tasks, for your smiles and positive mood.

I am thankful to all my friends in Finland, in Russia and elsewhere in the world for being curious about my life, work and studies. Your questions to me have often been a chance to re-evaluate the essence of my activities from another angle.

My parents, Nadezda and Alexander, and my sister, Natalia, regardless of the distance that separated us, have always been near supporting each of my steps in life. I am extremely grateful to you for growing me up, for your endless parental care, your love, constant help and generosity. Thank you for supporting and encouraging me in all my activities ever since my birthday. I owe you a lot.

Finally, I devote most of my thanks, appreciation and love to my darling wife, Ekaterina. Over the past years you have become the closest and the dearest person to me. Thank you infinitely for your endless patience, true love and tender care. I am very happy with you.

Espoo, August 2010
Helsinki, January 2011

Contents

Author's contribution

The major part of the results presented in this thesis has been published in five scientific articles [66, 67, 69, 78, 122]. 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] [69] was the first publication on evaluating feasibility of running IP security and mobility mechanisms, 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 analysed 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] [66] 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 smartphones required considerable time and effort. The OpenHIP implementation was ported and measured by Dmitriy Kuptsov. Together with him, we performed HIP experiments and collected measurement results that were then

[1] © 2007 ACM. 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), pages 1-8. ACM, New York, NY, USA. ISBN 978-1-59593-784-8.

[2] © 2009 IEEE. 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), pages 1-6. IEEE.

analysed by all the co-authors. I was the primary writer of the article's text except for few 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] [78] 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 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.

I co-authored the article *Elliptic Curve Cryptography (ECC) for Host Identity Protocol (HIP)*[4] [122] that proposes to replace RSA and DH public-key cryptography components in HIP with their ECC counterparts (ECDSA and ECDH) and analyses experimentally the benefits of such substitution. While the first author measured performance of ECC primitives on a server and estimated the number of HIP clients it can serve, my role in this work was to measure and analyse HIP without and with ECC on a resource-constrained PDA, Nokia N810 Internet Tablet. In addition, I contributed to the background chapter on HIP and ECC, estimations of ECC gains in HIP for Nokia N810, as well as participated in revising the whole article text.

In the article *On Application of Host Identity Protocol in Wireless Sensor Networks*[5] [67] we propose to use HIP as the main building block for network-layer security in wireless sensor networks (WSNs). We discuss how HIP can strengthen security of WSNs, suggest possible alternatives to its heavy components in particular WSN applications and evaluate their computational and energy costs on a Linux-based *Imote2* wireless sensor platform. I made a substantial effort to prepare our target device for experiments including flashing an Imote2 sensor with a custom Linux kernel, cross-compiling HIPL and OpenSSL code for it and setting

[3] © 2009 IEEE. 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), pages 1-9. IEEE.

[4] © 2010 IEEE. O. Ponomarev, A. Khurri, and A. Gurtov. 2010. Elliptic Curve Cryptography (ECC) for Host Identity Protocol (HIP). In: Proc. of the 9th International Conference on Networks (ICN 2010), pages 215–219. IEEE.

[5] © 2010 IEEE. A. Khurri, D. Kuptsov, and A. Gurtov. 2010. On Application of Host Identity Protocol in Wireless Sensor Networks. In: Proc. of the 7th IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS'10), pages 358–365. IEEE.

the IEEE 802.15.4 radio to work. I then measured performance of HIP and several cryptographic primitives on the Imote2 platform and evaluated the results in the article. I was the primary writer of the article's text excluding the chapter on lightweight alternatives (which is entirely a contribution by Dmitriy Kuptsov) and the chapter discussing potential benefits of HIP in WSNs (which is our joint work with Dmitriy). In addition, Dmitriy helped me with setting up the Imote2 radio interface, participated in energy consumption measurements and suggested to analyse different WSN applications. The analysis of WSN applications using HIP was our co-operative work.

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
BEX	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
DNS	Domain Name System
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
ESP	Encapsulating Security Payload
FQDN	Fully Qualified Domain Name
GNU	GNU is not Unix
GPRS	General Packet Radio Service
GPS	Global Positioning System
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
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
IRTF	Internet Research Task Force
KDF	Key Derivation Function
LAN	Local Area Network
LHIP	Lightweight HIP
MIP	Mobile IP
MOBIKE	IKEv2 Mobility and Multihoming
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
PKC	Public Key Cryptography
PKI	Public Key Infrastructure
POSIX	Portable Operating System Interface
RFID	Radio Frequency Identification
QoS	Quality of Service
RAM	Random Access Memory
RFC	Request for Comments
RFID	Radio-frequency Identification
RSA	Rivest-Shamir-Adleman
RTT	Round-Trip Time
RFC	Request for Comments
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
WSN	Wireless Sensor Network

List of Figures

List of Tables

1 Introduction

In this thesis we evaluate feasibility of using existing IP security mechanisms on lightweight devices with limited hardware resources and capabilities. In the following sections we first introduce to the reader the nature of the research problem and define our motivation for this work, and then specify contribution of the thesis and describe its organization.

1.1 IP technology goes embedded, Internet goes mobile

Migration of mobile telecommunication systems and small communication devices to IP technology is a growing and well recognized trend. While using different connectivity standards on the lower layers of the protocol stack (e.g., IEEE 802.11 and 802.15), today's mobile and wireless embedded devices do increasingly rely on a set of existing TCP/IP communication protocols on the upper layers to interconnect to each other and transfer data. However, by reusing existing technologies, the "mobile Internet" (as we call this ecosystem of interconnected embedded and wireless devices) also accumulates numerous security issues. Moreover, the scale of security threats in the mobile environment, which has properties and requirements that are different from the traditional Internet, dramatically increases. Existing solutions that have proved to work on conventional computers, may simply not be suitable for their mobile counterparts. Compared to wired and stationary communication systems, the mobile Internet i) is more prone to security attacks; ii) has less capabilities to employ strong protection mechanisms against these attacks and iii) does require that terminals support secure mobility, i.e. are able to change their location in a secure manner and seamlessly for applications.

Several research projects have successfully adapted existing Internet protocols to mobile systems resulting in a miniaturized TCP/IP stack perfectly suitable for an embedded environment with only few hundreds bytes of available memory [22, 24]. However, these solutions so far concentrated on the communications performance and general feasibility to run IP on tiny objects rather than on providing support for reliable security and mobility mechanisms. Mobility has not been in focus during the design of the original Internet, whereas perfect network-layer security in most cases still relies on strong cryptography. The latter involves computationally intensive operations that might easily stress resource-constrained devices and shorten the lifetime of applications running on them. A great variety of devices and ap-

plication scenarios with different security and performance requirements complicate the situation even further and produce a need for adaptable and flexible security systems.

Within the context of this thesis, we call "resource-constrained", "lightweight", "low-power" and sometimes "embedded" those communication devices with "limited" or "restricted" capabilities that are connected wirelessly, powered by a finite power supply (a battery) and typically have a small form factor, limited computational power, bandwidth and storage size compared to conventional laptops and desktop computers. Of these, devices such as PDAs, cellular phones, internet tablets, communicators etc. are usually used on the go, i.e. are mobile. Devices such as sensor nodes do not necessarily require support of mobility (though there are wireless sensor network applications requiring mobility of nodes) but keep valid other properties of the "lightweight" notion, while typically being even more constrained in resources. The target devices in our experiments have the processor clock rate ranging from 220 to 416 MHz (though the Imote2 sensor platform can operate in a low power, 13 MHz, mode), the amount of RAM ranging from 32 to 96 MB, data transfer rate limited by IEEE 802.11g and 802.15.4 standards and the rated battery capacity ranging from 860 mAh (for a mobile phone) to the capacity of three alkaline AAA batteries (for a sensor). As an additional class of "lightweight" devices presented in this thesis, we experiment with small network components such as OpenWrt-based 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's 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. The dual

role of an IP address, that due to the original design serves as both an identity and a topological locator for a host, is being considered as one of the biggest vulnerabilities of the current Internet [119]. Despite many proposed security mechanisms, it is not obvious how efficient they are on lightweight mobile devices and wireless sensors in the presence of limited computational, memory, bandwidth and energy resources.

1.3 Thesis contribution and scope

Our key contribution in this thesis is evaluation of the applicability of IP security to lightweight devices such as smartphones, Internet tablets, wireless sensor nodes and Wi-Fi access routers. In other words, we assess how feasible it is to run the existing network-layer security mechanisms on these classes of devices. Our interest lies especially in cryptography that is often a major part of a security architecture intended to authenticate mobile Internet hosts, protect confidential information transferred between them and defend from a variety of attacks. Operations such as generation of a shared secret key, encryption of data and public key signatures implemented in software, are computationally-intensive and could stress CPU and battery resources of a lightweight device. Potential consequences can be a long communication delay, a complete unresponsiveness to user input and a short lifetime of applications (due to fast battery depletion).

Improvements to cryptographic primitives and mechanisms as such are out of scope in this work. Instead, we evaluate suitability of an existing security protocol for lightweight hardware, identify scenarios where the protocol can be used unmodified and show how the protocol can be adjusted by using alternative security building blocks in order to meet operational and performance requirements of resource-constrained devices.

Host Identity Protocol

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 [59, 80, 81, 97, 98, 109, 110]. 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, secure mobility and multihoming. HIP uses public key cryptography (PKC) signatures to authenticate peers, a version of the Diffie-Hellman (DH) protocol to create a session key,

and can use IPsec Encapsulated Security Payload (ESP) to encrypt data packets. In addition, HIP uses a number of mechanisms to protect communicating parties from man-in-the-middle (MitM) and DoS attacks [98].

HIP defines a new namespace, the Host Identity namespace, consisting of cryptographic host identities being public keys of the hosts. The use of cryptographic identities for host identification allows HIP to split locator and identifier roles of IP addresses and, along with PKC signatures, to provide an elegant solution for securing identity and mobility. This property of the protocol is especially important for lightweight mobile Internet hosts that are the scope of this work. While the main focus of this thesis is the evaluation of network-layer security for resource-constrained devices, a part of our research is devoted to secure IP mobility, which, in turn, can be considered as one of the critical applications for such devices. We provide to the reader a background on IP mobility in Chapter 2, introduce related work in Chapter 3 and present our measurement results of mobility on a Linux PDA in Chapter 5.

A good property of HIP from the research perspective is that it relies on public key cryptography and IPsec. This makes HIP similar to other protocols and, thus, on the one hand, 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 and provides a way to split identifier and locator roles, the integration of HIP with other security protocols and systems becomes easier and even desirable [37, 119]. With all its advantages, HIP is a prospective candidate to be an essential part of the future secure mobile Internet architecture.

Performance evaluation on lightweight hardware

Although several previous projects evaluated HIP on standard Internet hosts [38, 49, 51, 60, 112], none of the studies has targeted a HIP assessment on a lightweight device with restricted resources. In this work, we perform measurements of security components of HIP on Linux-based Nokia Internet Tablets, Symbian-based smartphones and Imote2 wireless sensor motes. An extensive analysis of the results allows us to provide a set of recommendations on using unmodified HIP and IPsec on lightweight clients in varying circumstances. HIP is still an experimental protocol and not deployed widely. To move HIP specifications to the standard track and to trigger its large-scale deployment, HIP needs more practical knowledge and examples of its successful real-life usage. Therefore, our experience with HIP reported in this thesis is of particular importance to the research community working on the

future Internet. A survey of related work in the fields of IP security and mobility presented in the thesis gives to the reader a broader picture on the research problem and allows us to draw parallels with our own research results.

Our experimental results presented in Chapters 5–7 show that the applicability of HIP as a suit of security mechanisms to lightweight hardware depends on many circumstances including but not limited to the type of the device, specifics of the network environment and security and operational requirements of the application. In a hostile network with a high likelihood of attacks, where there is a need for applying stronger security measures (e.g., larger cryptographic key size, frequent rekeying etc.), or in a scenario with the need for multiple secure connections and mobility events in parallel, HIP turned to be too heavy for tiny communication devices.

To increase performance of applications in such cases while keeping security on a sufficient level, we suggest to replace selected "standard" HIP components with their lightweight alternatives that we present in Chapter 8. The alternative components include Elliptic Curve Cryptography (ECC), lightweight certificates, a polynomial key exchange and a recently proposed "HIP Diet Exchange (DEX)". For three examples of wireless sensor network applications, we provide our initial estimations on the level of gains from applying lightweight security mechanisms instead of original HIP components on the Imote2 sensor platform. Finally, in Chapter 9 we summarize cases where existing IP security mechanisms (i) can be used unmodified and (ii) should be tuned/substituted for resource-constrained devices.

HIP-based authentication architecture

A HIP-based distributed authentication architecture that addresses important security and mobility issues in wireless local area networks is another contribution of this thesis and can serve as an example of HIP application in real world scenarios. To validate our model we build a pilot WLAN with two models of Wi-Fi access routers with varying hardware resources. To further contribute to the subject of the thesis, we evaluate performance of selected public-key operations on these Wi-Fi access routers. This assessment allows us to determine a set of criteria for selecting equipment capable of running various IP security mechanisms and to set grounds for deployment of such an architecture in real-life wireless networks.

Porting experience

As an additional contribution, we present our first-hand knowledge about migration of open source software (OSS) to mobile environment, which can be useful to similar porting projects. In particular, we report our porting experience of two different HIP implementations to Symbian OS, as well as migration of HIPL (HIP for Linux) to embedded Linux and OpenWrt platforms. Our Symbian ports, though presently outdated, are available as a separate branch of the existing protocol implementations. On embedded Linux devices, the general HIPL code base can be used but needs to be cross-compiled.

Potential use of location information

Mobile terminals, wireless sensors and Wi-Fi access routers, which are the main target devices for evaluating IP security in this work, are increasingly utilizing information about their location. Although this is not the direct focus of this thesis, we consider the use of location information being an important subject relevant to our work on lightweight devices. Many existing examples of using location information for different purposes make it evident that it will play an essential role in the future mobile Internet both for enhancing functionality of different Internet protocols and for provisioning of location based services and applications. For instance, the Secure Mobile Architecture (SMA) presented in 2004 [25] and later developed and deployed by Boeing as a pilot infrastructure in the Boeing Intranet, integrates several components including a public key infrastructure (PKI), a directory service, HIP and location based network services [118]. Based on a set of predefined policies, SMA combines location and identity information of a user to determine her access rights to perform certain tasks and actions. SMA allows its users to operate not only within a designated factory area but also in the public Internet based on temporary certificates, secure identities and known location [118].

In wireless sensor networks, knowledge of sensor location or deployment information can be used, for example, to implement secure broadcasting, routing and access control mechanisms [140], as well as for optimizing key establishment schemes [86] and minimizing communication overhead during one-way hash function-based authentication of sensor nodes [21].

One of the key issues of utilizing location in the future mobile Internet will be to keep it secure. This should be done by incorporating location into security protocols and architectures, as it is done, for instance, in SMA. The approach of decoupling

identifier and locator roles of IP addresses used in HIP can be extended further to provide a mapping between a cryptographic identity of an object and its geographical location information obtained with distinct technologies (e.g., with GPS, network-based and sensing-based positioning). Our previous work [68] provides a certain basis for the use of location information on mobile devices by presenting the status of existing positioning techniques and by looking at various economical, social and privacy aspects that affect evolution of services relying on location.

Summary of contributions

To summarize, the key contributions of the dissertation are:

- Performance measurement results of HIP and selected cryptographic primitives, their analysis and identified limitations on resource-constrained devices;
- Evaluation of a set of potential protocol modifications and mechanisms that can overcome above limitations and make HIP suitable for lightweight hardware; analysis of WSN application examples;
- Protocol implementations enabled on embedded Linux and Symbian devices;
- Survey of related work in the fields of IP security and mobility;
- HIP-based authentication architecture for wireless LANs.

1.4 Thesis structure

The structure of the thesis is organized as follows. In *Chapter 2*, we overview related technologies from IP security and mobility fields, as well as present a background on the Host Identity Protocol that has been the target of our empirical research.

Chapter 3 surveys several interesting pieces of related work in the fields of IP security and mobility, security in wireless local area and sensor networks.

Chapter 4 describes the nature of the research problem and research methodology presenting research methods and a number of measurement and analysis tools.

In *Chapter 5*, we present performance evaluation of HIP on two types of mobile devices, a Nokia Internet Tablet and a Symbian smartphone. Section 5.1 contains description of our port of HIPL to a Nokia 770 and protocol performance results

including components of the HIP handshake and mobility update, power consumption of different operations, communication latency and data throughput with ESP encryption. Section 5.2 describes our experience of running HIP on Symbian OS and evaluates feasibility of IP security on lightweight cellular phones. We measure and analyse duration of the HIP base exchange and its parts, CPU load, RAM utilization and power consumption of protocol operations on Nokia E51 and Nokia N80 phones. The section also elaborates on the porting process of two HIP implementations, HIPL and OpenHIP, to Symbian OS.

Chapter 6 presents our HIP-based authentication architecture for wireless LANs and reports performance results of its components running on two Wi-Fi AR models, La Fonera FON2100 and Gateworks Avila GW2348-4.

Chapter 7 describes how HIP can be applied in wireless sensor networks and presents our performance measurements and estimations of energy consumption of numerous cryptographic primitives on the Linux-based Imote2 sensor platform.

In *Chapter 8* we discuss several existing security mechanisms that can make HIP a more lightweight protocol and, thus, better suitable for resource-constrained devices in particular application scenarios. We present measurement and estimation results on the benefits of these approaches.

We discuss the applicability of original and modified HIP to lightweight hardware, the need for adaptable security protocols and potential future research directions in *Chapter 9*.

Finally, *Chapter 10* concludes the thesis with a summary of results.

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 areas, terms and techniques.

2.1 IP security

The term "IP security" in its general meaning refers to a set of security mechanisms protecting network-layer (or IP layer) communications between the hosts in a communication system. Important components, processes and principles of IP security include but are not limited to authentication, authorization, privacy, confidentiality, integrity that altogether aim at defending against Internet attacks by means of different protocols and algorithms. The abbreviation "IPsec" stands also for IP security but covers a suit of concrete security protocols such as Authentication Header (AH) [63], Encapsulating Security Payload (ESP) [64], Internet Key Exchange version 2 (IKEv2) [62] and a number of other algorithms [65].

Cryptography mechanisms in communications systems are used to ensure data integrity and confidentiality by encrypting 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 [18, 20, 125, 130]. 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 that is a specific piece of information. In *symmetric cryptography*, this key is either 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 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 [18]. Examples of the most known block cipher methods used with symmetric cryptography are DES (Data Encryption Standard) [100], AES (Advance Encryption Standard) [101], and Blowfish [128].

2.1.2 Public-key cryptography

Public-key cryptography (PKC), also known as *asymmetric cryptography*, uses an approach different from symmetric cryptography. The main idea is built around a pair of the keys, public and private, that are bound to each other mathematically 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 easily derived from its public counterpart [20, 125].

PKC allows not only to preserve communication privacy between a host and its peers by making it impossible to intercept information by a third party 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 [125]. A downside of PKC is that the algorithms it uses operate on large prime numbers, involve computationally intensive operations 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 communicating parties is a lightweight host with limited computational capabilities. In practise, asymmetric and symmetric cryptography algorithms are often utilized jointly. First, PKC 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 [18].

RSA

RSA algorithm [125] 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 PKC. 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 need for 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 (this data may include, e.g., a symmetric key), while signing the messages with the private key authenticates the host to its peers. More details about the RSA algorithm can be found in [125].

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 [103].

Diffie-Hellman

Diffie-Hellman (DH) [20] 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 [20].

2.1.3 Elliptic Curve Cryptography

An approach to use elliptic curves in cryptography as an alternative to the existing public-key algorithms (originally to the Diffie-Hellman protocol) has been proposed by Victor Miller in 1985 [94]. 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., [71, 72]). Along with Miller, he is referred to as one of the ECC inventors. The strength of ECC lies on the elliptic curve discrete logarithm problem (ECDLP), i.e., the complexity of finding discrete logarithms. In turn, the efficiency of ECC is due to efficient scalar multiplication of elliptic curve points [16]. Largely for this reason, ECC is often being called as a "new generation" of PKC.

Although with present requirements to security levels it does not bring much efficiency comparing to the "mainstream" algorithms such as DH or RSA, ECC will provide significant benefits for future communications systems when security requirements will be multiplied from their current level. In other words, while increasing the security level, computational cost per bit (of the key length) grows with ECC much slower than with other PKC mechanisms [105]. Besides computational efficiency, an important aspect of ECC is that it requires notably shorter key length than RSA and DH while preserving an equivalent security level (see Table 2.1). This decreases the required code and memory space, as well as the size of messages transmitted over the air. Presently used RSA/DSA cryptosystems still often employ 1024-bit keys, which was considered to remain "unbreakable" until 2010 [16]. Already now many services in the Internet performing critical transactions do increase the security level by using, e.g., 2048-bit RSA keys, and this is the time where ECC will start being more beneficial for communication systems.

Looking at the current ECC implementation and deployment status one can observe certain implications. The biggest obstacle in applying ECC on a large scale at a fast pace, which substantially differentiates it from the rest of cryptography methods, lies in numerous patents that cover the majority of ECC-related techniques and algorithms. One notable player in this area is a Canadian company *Certicom* that holds over 100 patents concerned to ECC and PKC in general [105]. According to

Table 2.1: Comparable key sizes with different cryptosystems. Adapted from [16] and [17].

Security level	**Symmetric**	**ECC**	**DSA/RSA**	**"Best Before"**
80	80	160	1024	2010
96	96	192	1536	N/A
112	112	224	2048	2030
128	128	256	3072	2040
192	192	384	7680	2080
256	256	512	15360	2120

Certicom[6], 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[7], which gives an opportunity to solve the ECDLP at its current level practically to anyone. 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 of 109-bit. The 131-bit challenge requires much more resources to be solved, whereas stronger challenges (starting from 163-bit) are treated as computationally infeasible.

Besides mentioned activities, Certicom has formed the Standards for Efficient Cryptography Group (SECG)[8] that combines core providers of cryptography solutions, seeking for interoperability between them. Among other, the SECG has been producing a number of ECC working drafts and standards including "SEC 1: Elliptic Curve Cryptography, Version 2.0" [16] and "SEC 2: Recommended Elliptic Curve

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

Domain Parameters, Version 2.0" [17]. Certicom classifies ECC algorithms into three categories[9]: (i) the algorithms for digital signatures including Elliptic Curve Digital Signature Algorithm (ECDSA), Elliptic Curve Pintsov Vanstone Signatures (ECPVS) and Elliptic Curve Nyberg Rueppel (ECNR); (ii) the algorithms for key negotiation including Elliptic Curve Menezes-Qu-Vanstone (ECMQV) and Elliptic Curve Diffie-Hellman (ECDH); and finally (iii) the Elliptic Curve Integrated Encryption Standard (ECIES) for encryption. Of the above, in this thesis we evaluate performance of ECDSA and ECDH that are specified in "Standards for Efficient Cryptography 1 (SEC 1)" [16], as well as in FIPS PUB 186-3 [103] and NIST SP 800-56A [102] standards.

Section 3.1 of this thesis surveys some relevant studies on ECC security on mobile and wireless sensor devices, whereas in Section 8.2 we report our measurement results of selected ECC primitives on Nokia N810 Internet Tablet and Intel Mote 2 sensor platform.

2.2 IP mobility

Mobility is one of the essential attributes of today's Internet users and networks. Especially, the owners of lightweight portable 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 address. Other mobility 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* [75, 88, 129, 148].

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 [88, 129, 148]. *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 [19, 88, 148]. *Flow mobility* allows splitting parts of a connection

[9] http://www.certicom.com/index.php/ecc-based-algorithms

between distinct interfaces of a host, as well as switching a communication between IPv4 and IPv6 protocols [148]. *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 [129]. *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 [129]. 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 [129].

2.2.2 Handover types

A comprehensive set of mobility related definitions can be found in [88]. 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 [88]. 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 [88].

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 [14, 58, 73, 75, 109, 114, 120, 135, 147, 148]. 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 [133, 142, 143].

In this work, we are mostly concerned with secure IP-layer mobility that, in terms of the above 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) [37, 59, 80, 81, 97, 98, 108, 109, 110] 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).

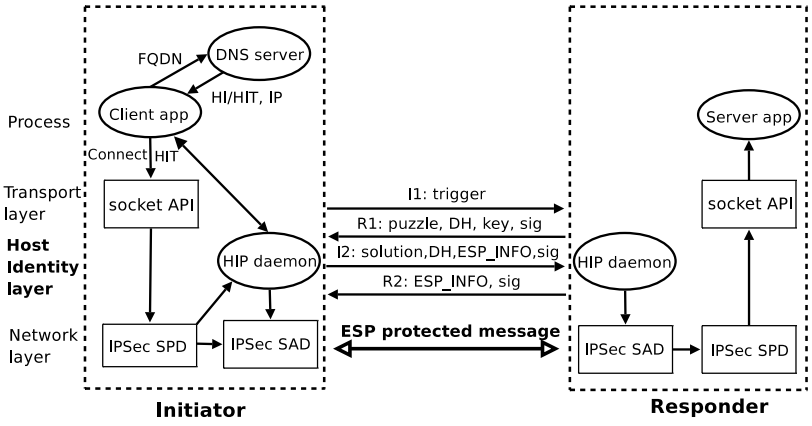


Figure 2.1: HIP architecture.

HIP offers several benefits including end-to-end security, resistance to CPU and memory exhausting denial-of-service (DoS) attacks, NAT traversal, secure 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) [97, 98] 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 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) [111] 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 BEX. The initiator can retrieve the HI/HIT of the responder from a DNS directory [110] 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 IP address by

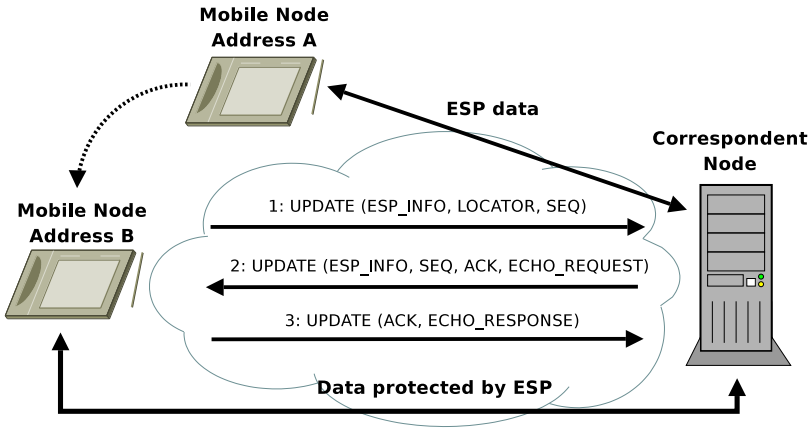


Figure 2.2: HIP mobility update. Adapted from [109].

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 signalling mechanism between communicating nodes (see Figure 2.2) [109].

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 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 [109].

In some cases a host might have more than one IP address 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 [109]. Several extensions have been proposed over past years in research to complement HIP with micromobility [114, 134, 149] and network mobility [52, 115] support. Some of these studies are mentioned under related work in Section 3.2.

3 Related Work

A number of studies evaluated different network-layer security and mobility mechanisms running on traditional computers [49, 60, 148]. Several research projects studied security aspects of communications in wireless sensor networks [41, 85, 121, 126, 137, 141]. Other studies were devoted to protection of sensitive health care data being transferred between remote monitoring mobile devices [87]. This section highlights interesting related research on IP security and mobility, focusing on lightweight mobile devices and sensor motes.

3.1 Research on IP security

In the following sections we survey several research studies that aim at securing IP layer communications of mobile devices by means of symmetric and asymmetric cryptography including ECC. We also overview the Internet Key Exchange protocol and its mobile extension. As a part of our research on evaluating IP security on lightweight hardware, we have assessed applicability of HIP to wireless sensor motes and analysed the impact of its security components on wireless sensor network (WSN) applications with different requirements. The respective details of this research are presented in Chapter 7. A considerable part of related projects presented below deal with wireless sensor networks. Numerous approaches have been suggested to mitigate potential attacks on security in WSNs. Nevertheless, the survey shows that many issues of WSNs remain open and call for adaptable security protocols that can be utilized in various scenarios and circumstances.

3.1.1 Security in mobile systems

Malhotra *et al.* [87] 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 provides data encryption and user authentication. The authors emphasize the great advantage of ECC over RSA on resource-constrained mobile devices that comes from shorter ECC key lengths resulting in lesser occupied memory and faster processing time. As an example, a 1024-bits RSA key's size would be equivalent to the size of a 160-bits ECC key [87]. Unfortunately, besides referring to other studies in the ECC field, the authors do not provide empirical comparison 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 $SPV\ M5000$ PDA 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 produces 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) [87].

He and Zhang [42] 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 [42].

3.1.2 Security in wireless sensor networks

A number of projects have studied security issues in WSNs. Liu and Ning [85] discuss the problem of establishing a pairwise key in sensor networks. While claiming infeasibility of using PKC for such purposes on resource-constrained sensor nodes, the authors propose an alternative key predistribution framework based on polynomials and analyse two schemes derived from this framework. In Section 8.4 of this thesis we apply a similar to [85] scheme, as well as ideas from [34] to HIP. The same authors in another study [84] describe design, implementation and evaluation of the *TinyECC* library for applications running on TinyOS-based sensors. Besides the library code, the authors provide measurement results that clearly indicate performance of selected ECC primitives on four sensor platforms. However, as the authors

note, without additional security mechanisms, TinyECC is susceptible to DoS attacks that can deplete sensors' batteries [84]. This conclusion exemplifies the need for a combination of techniques to achieve better security in WSNs. Making a parallel to HIP, it has a built-in puzzle mechanism intended to protect HIP responder against potential DoS attacks on the network layer by enforcing a potential attacker to spend CPU cycles for solving a cryptographic puzzle. However, the appropriateness of this mechanism for sensor networks should be considered carefully. Our preliminary analysis in Section 7.3 indicates the lack of WSN scenarios where the HIP puzzle mechanism would be worth using.

An extensive survey of security requirements, threats and corresponding defensive measures is presented by Walters *et al.* [140]. The survey provides a solid reference base for further research on WSN security. Piotrowski *et al.* [121] provide an estimation for the lifetime of different sensor models running PKC primitives. After comparing RSA and ECC algorithms, the authors conclude that the use of ECC is feasible even on the weakest of the tested motes, MICA2DOT.

Roman *et al.* [126] survey a wide combination of cryptographic primitives implemented in software and hardware and evaluate their applicability to different wireless sensor models, ranging from "weak" PIC12F675 (uPart) motes to "heavy-duty" ARM920T and PXA271 (Imote2) devices. As the authors point out, despite numerous promising research efforts on implementing asymmetric cryptography algorithms, in particular ECC, on microcontrollers or chips, most of the existing techniques to date are just proof-of-concept. On the other hand, already implemented in hardware AES-based cryptography (e.g., on chips CC2420) according to Roman *et al.*has certain flows and restrictions, such as proneness to replay, DoS and forging attacks (AES-CTR encryption in counter mode) and limited amount of entries in an ACL (e.g., two entries in AES-CCM (AES counter message authentication code with cipher block changing mode)). Regarding software-based primitives, the authors conclude that while tiny sensor nodes are almost comfortable to perform ECDSA signature and verification operations, advances sensor models have sufficient resources to handle any kind of cryptography, either symmetric or asymmetric. Overall, the authors provide a broad picture of performance of different cryptographic primitives and pay attention to the balance between time, memory and energy resources consumed by them. This uncovers a number of potential optimizations that can improve suitability of cryptography for different WSN platforms [126].

Wang *et al.* [141] present a comparison between symmetric cryptography and ECC-based PKC in sensors networks. The main objective of the 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, 128-KB flash memory and 4-KB 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. The conclusion is that ECC-based PKC has more advantages over symmetric cryptography in terms of message complexity, the use of the memory and security [141].

A recent study by Haque *et al.* [41] 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 the 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 of the systems, 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 PKC) 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.

A study by Nachman *et al.* [99] is completely devoted to the Imote2 sensor platform presenting its technical advances over previous sensor models and introducing potential Imote2 applications. By running a set of measurements of public-key cryptography while ranging CPU clock frequency, the authors draw a conclusion that an

advanced Imote2 sensor device consumes less energy for an ECC key generation than a predecessor *TelosB* only when the CPU of the former is clocked at high frequencies (above 208 MHz). Sun *et al.* [137] design an access control system for WSNs that consists of an ECC-based admission module (combined with a polynomial authentication), an Advanced Encryption Standard (AES)-based hardware security interface implemented in *CC*2420 radio components on TinyOS and a scheme for updating the network-wide symmetric key. Implemented on Imote2, the system includes the components similar to our prototype.

Comparing to related research, our work on WSNs is different in a number of ways. First, while many studies assess performance of individual cryptography primitives for particular sensor platforms, we show how a standardized secure communication protocol can be applied in WSNs. Second, while acknowledging suitability of PKC for WSNs shown by previous research, we provide measurements of computational and energy cost of the protocol components performed on a real Imote2 sensor platform that is running Linux unlike many TinyOS-based devices used in the previous projects. We emphasize that Linux has a great potential for future generations of sensors. Finally, we recognize the vast difference in hardware resources of sensor devices and WSN application requirements and accordingly do present and evaluate several lightweight security mechanisms that are able to enhance computational performance and prolong the lifetime of a WSN in particular scenarios, while providing sufficient security level.

In general, existing research on applying cryptography in mobile and wireless sensor networks provides a solid basis for future work and, most importantly, for implementation of security systems intended to protect real applications. However, related work shows that, although most of the cryptographic primitives have been evaluated as individual components on different mobile and sensor devices, a combination of such techniques is often needed in practice to build a multi-purpose security service adaptable to requirements of various applications. To this end, there is little knowledge on applicability of existing security protocols to such hybrid security schemes to be used on resource-constrained devices. This thesis aims to provide new insights into this area by surveying related work, analysing different combinations of existing security mechanisms and evaluating their performance on several types of lightweight devices.

3.1.3 IKE and MOBIKE

Internet Key Exchange version 2 (IKEv2) protocol is standardized in RFC 4306 [62]. 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 ESP and AH protocols [64, 63]. 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.* [83] 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 [70] 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/addresses 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 [98]. 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 [56].

3.2 Research on IP mobility

In this thesis, besides presenting our practical experience with HIP mobility, we find it necessary to look at the related studies and to overview some essential approaches that make host mobility and multihoming possible. 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 specific 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 5.1.2 indicate that secure mobility produces certain costs to a mobile handset in terms of computational 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

Mobile IP (MIP) [120] 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 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 [120].

In research, there can be found several comparative studies and performance evaluations of different mobility protocols on conventional PCs or laptops. Henderson *et al.* [51] 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 [58]. 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-centred approach; security of MIPv6 versus HIP mobility update mechanisms [51]. 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" [7]. 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 [109]. Besides this, Novaczki *et al.* [115] 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 [60] compare performance of a vertical handoff with HIP and MIPv6 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 an extremely valuable 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 [13, 36, 82, 61, 133, 142]. 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 [82, 61, 142] proposed a multilayered mobility management approach based on a combination of the Session Initiation Protocol (SIP) [127] and Mobile IP.

Schulzrinne and Wedlund [129] 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 [129].

Similarly with Mobile IP, HIP has become a candidate protocol to combine it with SIP for cooperative mobility management on two layers. Henderson [50] 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 [50]. 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 [114, 134, 149] and network mobility [115, 52] support.

Since the work by Henderson, HIP and SIP integration has been further reflected in research. A hybrid scheme called *SHIP* has been suggested by So *et al.* [133]. Based on experimental evaluation, the authors showed that SHIP outperforms a hybrid MIP and SIP scheme in terms of signalling overhead and efficiency [133]. A more

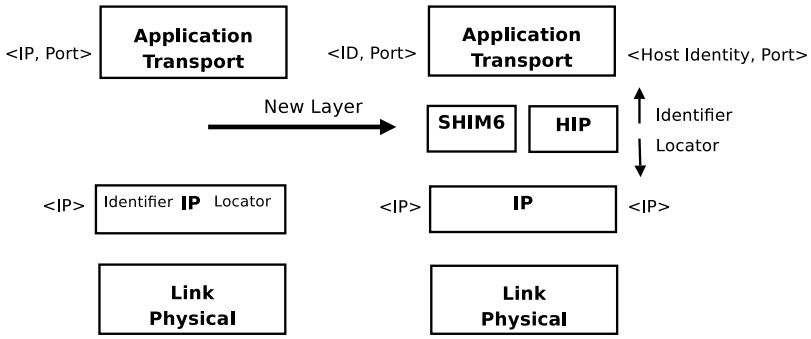


Figure 3.1: SHIM6 and HIP layers in the protocol stack.

recent work by Camarillo *et al.* introduces a framework for combining SIP and HIP and emphasizes its advantages [13].

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" [113]. 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 [113]. 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.* [51] present their performance results of HIP running on Dell Latitude laptops. These results are interesting to us in respect to our own empirical study (see Chapter 5.1)

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 [51] whereas the Nokia 770 is powered by a 220-MHz ARM processor [69]). Obviously, there are other potentially influential factors that, in fact, have been different in the experiments (e.g., processor architecture, Linux kernel version, network connection characteristics and the amount of memory). 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 mobility update time is spent on the cryptographic operations such as signing and verification procedures [51, 69].

Another study by Nikander *et al.* [112] 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 ten respectively. Further increasing of K raises processing time by a large factor, approaching 100 seconds when K amounts 15 [112]. This exponential growth goes very well with theoretical expectations, the results presented by Henderson *et al.* [51] and our own results illustrated in Section 5.1.2. However, in our mea-

surements on 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.* [112]. Potential reasons for result distinction might come from differences in the HIP implementations, operating systems and underlying hardware platforms.

Pääkkönen *et al.* in a recent study [117] 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 [117].

3.4 Security and mobility issues in wireless LANs

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 6. Our original work [78] has much in common with research problems described in this thesis because 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 lightweight components. Not all models of Wi-Fi

access routers, for instance, are equipped with sufficient hardware resources to become a part of such architecture. Section 6.4 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.* [45, 47]. 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 6. 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 [29] considers distributed authentication, authorization and accounting (AAA) in Wi-Fi community networks. It focuses on building trust chains between communicating entities using certificates and certificate authorities (CA). Later, valid certificates serve for authenticating clients, as well as for identifying and validating decentralized AAA servers.

The concept of a distributed firewall (used in our approach and presented in Section 6.2) is not novel but existed for a long time. Related studies [55, 136] 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, the authors present a distribution scheme and an authentication technique for network entities participating in the policy enforcement process.

Validation of source addresses of a communicating host is another topic related to our research. Source address validation architecture (SAVA) [144, 145, 146] addresses the problem of source address spoofing and accounting on different levels of granularity, from a local subnetwork to autonomous systems. Another architecture called SAVAH (SAVA with HIP) [77] complements SAVA with the Host Identity Protocol and cryptographically verifiable identifiers, allowing clients of a local network to be authenticated, authorized and accounted in a secure manner on a first-hop network router. While being an alternative to Cryptographically Generated Addresses (CGA), SAVAH provides support for both IPv4 and IPv6 and is free from patents [77].

4 Research Problem and Methodology

In this chapter we formulate the research problem of the dissertation and describe methods and tools used to conduct the research and experiments.

4.1 Research problem

In the following paragraphs we discuss several perspectives of the research problem and show how we are going to address it.

4.1.1 Security perspective

Although there have been several activities on adapting IP technology [24, 22] 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, memory and battery constraints of lightweight devices, as well as constraints of an underlying wireless network (e.g., limited bandwidth, frame size, low signal level) raise concerns whether IP security mechanisms might be employed there without major modifications. In particular, asymmetric cryptography algorithms implemented in software, which are employed to secure communications between mobile resource-constrained devices, can easily stress their CPU and memory resources, as well as negatively affect battery lifetime, data throughput and communication latency.

In this work, we intend to bring more awareness about applicability of IP security to lightweight devices by measuring and evaluating performance of different security components constituting the Host Identity Protocol. We analyse the impact made by the heavyweight cryptographic operations on CPU and RAM utilization, energy consumption and battery lifetime of mobile devices and wireless sensor motes. On the other hand, we also assess the efficiency of the protocol and general feasibility of IP security for lightweight hardware by measuring duration of certain protocol components including initialization, generation of a public-private key pair, puzzle solving procedure, HIP base exchange and HIP mobility update. In addition, we

evaluate the effect of IPsec ESP encryption (involved with HIP) on communication latency and TCP throughput. This allows us to make recommendations on suitability of unmodified HIP for lightweight mobile devices and sensor platforms.

Adaptable security

From another perspective, the analysis of applications with different security requirements running on devices with varying capabilities indicates that in certain scenarios (e.g., when there is a need for strong security in an extremely constrained environment or a need for multiple parallel security connections), unmodified HIP is too heavy. It becomes evident that modifications or adjustments to the protocol are necessary and that, to be suitable for different applications and devices, a security protocol has to be flexible and adaptable to varying circumstances. In this thesis, in addition to evaluating HIP in its original form, we assess possible lightweight alternatives to particular protocol components, aiming to decrease the overhead of security and raise the efficiency of the protocol on lightweight hardware. Our intention is to demonstrate different security components running on lightweight communication systems that, being able to suit different scenario needs and meet distinct operational requirements, can form a general and flexible security framework.

Secure mobility

Although the focus of the dissertation is network-layer communications security, we also consider other related issues particularly important for target devices of this work. Namely, we consider support of mobility as one of possible applications of a security protocol and as one of the additional benefits HIP can bring to devices without a fixed network attachment point. Efficient and reliable mobility solutions are necessary for such devices, as otherwise they would lose their notion of being *mobile*. Without mobility support, although a user can still carry the device with her, most likely she will not be able to utilize the full communications potential of the device (e.g., to keep Internet sessions ongoing due to a 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 must be secure.

Numerous mobility solutions proposed in the literature range from network and application-layer mobility to multilayered approaches [51, 112, 60, 61, 82, 133, 50]. However, little knowledge is available on how well these solutions scale to the nature of embedded platforms that have requirements distinct from desktop computers and often behave differently. A mobile phone changes its network attachment point more frequently than a laptop or a PC. In addition, limited resources of such a device may produce a need to adjust or tailor existing mechanisms to achieve better performance.

In the experimental part of this work, we measure efficiency of the HIP secure mobility mechanism running on lightweight communication clients with constrained resources and compare handover performance with more powerful devices. 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 duration of an update procedure to be able to sustain an IP address change.

4.1.2 Energy consumption perspective

An essential part of the analysis of applicability of IP security to resource-constrained mobile and wireless devices is to evaluate how efficient a protocol or its component is in terms of energy consumption. The concept of accumulating power from a small portable battery rather than from a fixed power outlet allows devices to be wearable, which partly constitutes the notion of their *mobility*. Challenges, however, arise from limited battery capacity that often prevents applications, especially involving wireless data transmission and intensive computations, 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 wired connection to an electricity supply. On the other hand, movement of a device is not the only limiting factor for recharging possibility. For instance, wireless sensor motes placed in a remote location are often physically inaccessible once they have been deployed, providing no opportunity to replace batteries. Thus, the lifetime of wireless sensor network applications is determined by the battery lifetime of sensor nodes forming the network. This necessitates optimizations of performance of a security protocol in terms of duration of both on-device operations and data transmission.

Data transmission over a wireless interface can consume more energy of a mobile

or sensor device than intensive computations running on it. To design an efficient security protocol, it is crucial to find a compromise between the volume of computationally intensive operations performed on the device and the volume of data sent over the network. As previous research shows, depending on the computational capabilities of a sensor device it can be worth to process as much data on board and to transmit as little data over radio as possible [99]. Potential ways to decrease amount of transmitted data is to compress it prior to sending to the network.

Coping with battery constraints on lightweight mobile handhelds and wireless sensors is not an easy task from the engineering perspective. For instance, increasing battery capacity cannot be a long-term solution to the problem, as it produces more heating to a handheld and makes it less comfortable to use. Moreover, the multitude of applications running on modern communication devices increases the gap between the amount of energy being consumed and innovations in the battery development, i.e. the gap between energy demand and supply. The use of wireless connection and computationally expensive cryptography operations on lightweight devices increases energy consumption and accelerates battery depletion.

One of our objectives in this thesis is to evaluate the impact made by certain components of IP security on power consumption and battery lifetime of a lightweight device. In our experiments, we measure power consumption of the Host Identity Protocol operations on mobile devices. For several types of wireless sensor network applications, we estimate energy efficiency of HIP using its standardized and alternative security components. Since our target devices are battery-powered, energy consumption is one of the important indicators of feasibility of running certain IP security mechanisms on such devices. One of the greatest challenges in building practical security services for lightweight communication systems is to find a trade-off between a minimum required security level and the maximum achievable lifetime for a system.

4.2 Methodology

In this section we present methodology for our work. After defining main research methods, we describe development environments, experiment setups and primary measurement tools and utilities that we used.

4.2.1 Research methods

This work used a combination of empirical and engineering approaches in research. Among different methods the dissertation primarily relies on experiments with real-world implementations, observations and evaluations. The main components of the work are literature study, development of protocol software implementation ports, measurements performed on real communication devices in pilot networks, and subsequent analysis of the results. Through the literature study we obtained the insights into the research problem, evaluated the contribution of the related work in the field, and gained necessary background knowledge. The survey of the related studies allowed us to draw parallels with our own empirical results. The empirical research for each type of hardware platform consisted of the following phases: implementation (porting of the protocol code to the platform in question, configuration and possible extensions), testing (validation of correct work and functionality), planning (definition of measurement scenarios and characteristics to measure), measurements (getting actual data for evaluation) and analysis (assessment of result implications, comparison of performance between different devices or against existing results in the literature).

The empirical results were conducted in a number of experiments 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 general network characteristics. To get statistically more accurate results we practised repetitions of measurements. We then performed an extensive analysis of the results from different perspectives and made conclusions about feasibility of using unmodified protocol components on resource-constrained communication platforms, as well as evaluated lightweight alternatives to particular protocol components to be suitable even for the most constrained environments and scenarios.

4.2.2 Research tools

This section presents a set of tools that we have been using to accomplish the empirical research part. In the following paragraphs, we first describe the development environment used to port HIP to Maemo and Symbian mobile platforms, and then introduce the main test network components followed by a list of measurement tools.

Development and testing environment

We performed our experiments on different desktop computers and mobile devices. The desktop computers we used ran a Linux Ubuntu distribution. To run our tests on mobile devices and measure their ability to run the security components of our interest, we first needed to make the respective software available on this devices. With respect to the HIP protocol, this required porting an existing implementation from desktop to mobile environment. In this phase (as we call it *development* phase), we ported existing HIP implementations to three types of mobile/embedded platforms, Linux Maemo, Symbian S60 and an embedded Imote2 Linux distribution.

For Maemo, we took the existing HIPL (HIP for Linux) protocol implementation and compiled it for Nokia Internet Tablets using the Scratchbox cross-compilation toolkit and a Maemo SDK. Further details on this process can be found in Section 5.1.1. For Symbian, the porting procedure has been far more difficult due to restrictions of public Symbian SDKs at that time and the Linux-dependent HIPL code. In this respect, the Boeing's OpenHIP implementation was more platform-relaxed and easier to port. However, 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 5.2.1. For Symbian development 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. We also employed the Symbian S60 Emulator supplied with the SDK, which allowed us to debug runtime code errors with less effort and to test protocol operations before measuring their performance on the actual mobile phones, Nokia E51 and N80. For Imote2, the HIPL migration and development work has been done on a desktop computer using a cross-compilation toolchain.

Pilot network environment

Measurements have been an essential part of our experiments to conduct data about performance of cryptographic primitives and other security components on different mobile and wireless sensor platforms. For experiments with mobile handhelds, 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. For experiments with the Imote2 wireless sensor platform, we formed a network consisting of several Imote2

sensor nodes each supplied with a battery and sensing board and communicating with each other over IEEE 802.15.4 radio interface.

Measurement and analysis 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 the *tcpdump* utility and the *Wireshark* network analyser to capture traffic, analyse individual data packets and record their arrival and departure times to calculate the duration of particular protocol operations.

To measure computational cost of different cryptographic algorithms and operations, we used the *OpenSSL speed* utility [10], a benchmarking program available in the HIPL software, as well as timestamps inside the HIPL code. The OpenSSL speed measurement utility was used for measuring duration of RSA, DSA and ECDSA signature operations, and ECDH key exchange. A separate utility was used to measure computation time of a DH shared key. In addition, we created a set of shell scripts to automate measurement process and to ease repetition of the measurements.

With the Nokia E51 smartphone, we utilized *Nokia Energy Profiler* [31], 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 further analysis. Profiling data in this tools includes information about power and memory consumption, CPU load etc.

With Nokia E60 and N80 smartphones, we used *Carbide.c++ Performance Investigator* [11] to collect and analyse data about usage of different resources. Performance Investigator consists 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.

To measure current draw (and then estimate energy cost of security primitives), we used an external multimeter. With the Imote2 sensor platform, it was a *Fluke 189* digital multimeter with a DC current accuracy of $\pm$ (0.15% + 2). The multimeter was able to calculate and display an average DC current draw with the minimum

[10] http://openssl.org/docs/apps/speed.html#

[11] http://wiki.forum.nokia.com/index.php/Carbide.c%2B%2B:_Performance_Investigator

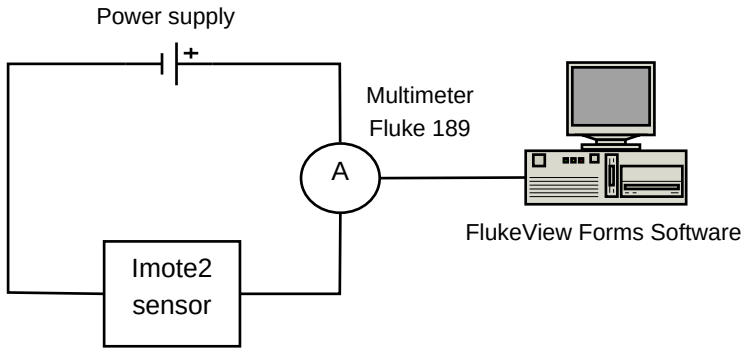


Figure 4.1: Current measurement setup.

1-second interval. For short events lasting for microseconds and milliseconds (e.g., RSA signature verification), we thus needed to do more repetitions in a row than for longer events, so that a sufficient number of multimeter readings for a particular event can be collected and analysed statistically. For our convenience, all multimeter readings were transferred and stored on a PC using the *FlukeView Forms* software. A schematic view of our simple current measurement setup is depicted on Figure 4.1. With Nokia 770 Internet Tablet, we used a less precise model of an external digital multimeter.

For statistical analysis of measurement results and their visualization we utilized several tools and application suits such as *Microsoft Excel* [12], *OpenOffice.org Spreadsheet (Calc)* [13], *Gnuplot* graphic utility [14] and *R* environment for statistical computing [123].

[12] http://office.microsoft.com/en-us/excel/
[13] http://www.openoffice.org/product/calc.html
[14] http://www.gnuplot.info

5 Evaluation of Host Identity Protocol on Mobile Devices

This chapter is devoted to HIP evaluation on mobile handhelds. In Section 5.1 we describe our experience with running the Host Identity Protocol on a Linux-based Nokia Internet Tablet and Section 5.2 contains performance evaluation of HIP on Symbian-based Nokia smartphones.

5.1 Performance of HIP on Nokia Internet Tablets

The current section is based on our article *Performance of Host Identity Protocol on Lightweight Hardware*[15] [69]. Section 5.1.1 briefly describes the Nokia 770 Internet Tablet hardware and software, as well as our port of the HIPL implementation. In Section 5.1.1 we present the components of our experimental testbed. Section 5.1.2 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 measurements and conclude about potential HIP implications for similar resource-constrained mobile devices. Finally, Section 5.1.3 summarizes our performance evaluation on Nokia 770 with a list of recommendations.

Our choice of Nokia 770 Internet Tablet as a target device for experiments 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 mechanism, in our case HIP mobility extensions. Next, at the time of the experiments, Nokia 770 was gaining its popularity among end-users and developers, which resulted in a number of created 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.

[15] © 2007 ACM. Portions reprinted, with permission, from 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), pages 1–8. ACM, New York, NY, USA. ISBN 978-1-59593-784-8.

5.1.1 HIP on the Nokia Internet Tablet

This section outlines the Nokia 770 technical specifications, describes the porting process of the HIPL implementation and present test environment for experiments.

Nokia 770 specifications

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 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 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.

HIPL porting process

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) [1], developed at Helsinki Institute for Information Technology. Although the HIP daemon and other utility programs of HIPL are 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.

Test environment

We performed our measurements on a Nokia 770 Internet Tablet (from now on - Tablet) and an Intel Pentium 4 CPU 3 GHz machine with 1 GB of RAM (PC)

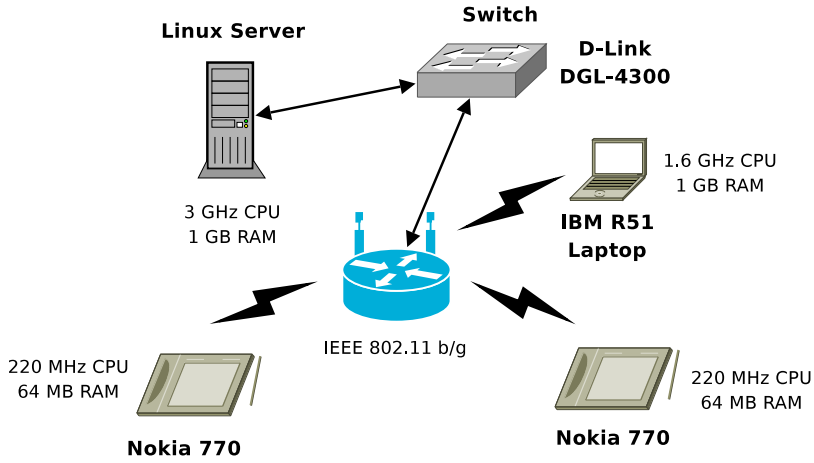


Figure 5.1: Test network with Nokia 770.

connected to each other via a switch and a WLAN AR in our test network (see Figure 5.1). 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 (Laptop) connected to the PC over the same wireless link as 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.

5.1.2 Experiment results on Nokia 770

This section presents the results of our experiments with the Host Identity Protocol and the IPsec protocol running on Tablet and Laptop.

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 Tablet in comparison with 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 5.2 depicts the times that were measured in our experiments. We leave 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 a pre-created and signed R1 message and adds a puzzle to it just before sending the packet to the 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 5.3 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 mobile devices, Tablet and Laptop. As the figure indicates, Laptop greatly outperforms Tablet for all operations involved with BEX. T2 time for Tablet is nearly 1.2 seconds, which is significantly longer than the respective one of Laptop (0.14 seconds). The majority of T2 is spent by 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 the 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 PC while Tablet acted as the Responder. The results suggest that the base exchange time is independent of whether PC or Tablet initiates the handshake. In both cases, with precreated R1

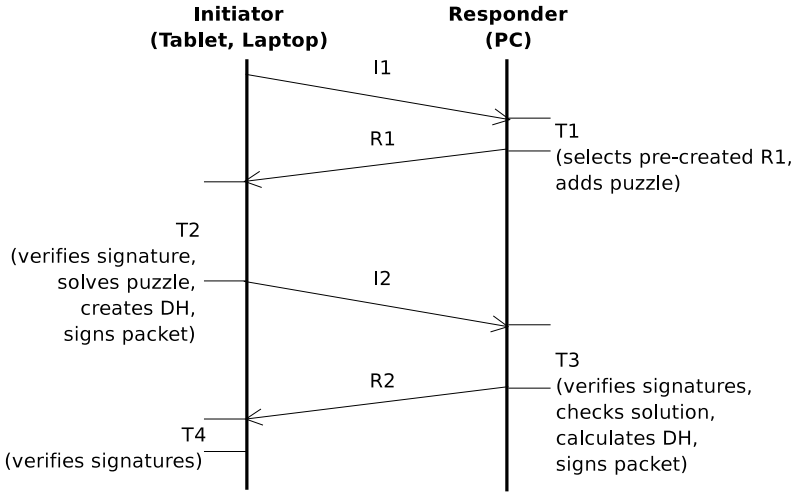


Figure 5.2: Time spans measured on the Initiator and the Responder.

packets, the base exchange lasts around 1.4 seconds.

Although 1.4 seconds to perform a HIP handshake between a mobile client and the correspondent server 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. 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 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 BEX time is exerted by cryptographic operations costly for Tablet's CPU. Such operations include generation and verification of signatures, as well as computation of the Diffie-Hellman session key.

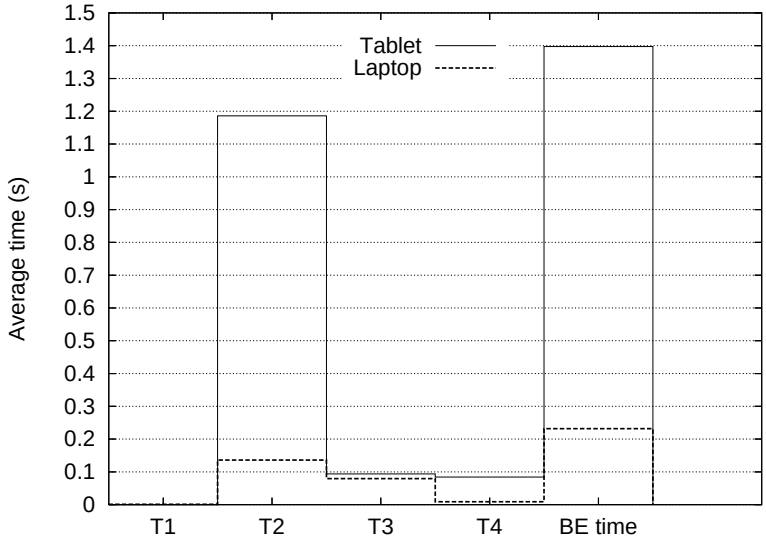


Figure 5.3: Duration of HIP base exchange stages for Tablet and Laptop.

Puzzle difficulty

Upon receiving an R1 packet, the Initiator is expected to solve a cookie challenge (a cryptographic 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 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 [98]. 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 5.4 illustrates this dependency for Tablet and 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 5.1 we present the mean values and the standard deviation of the T2 processing time that includes the

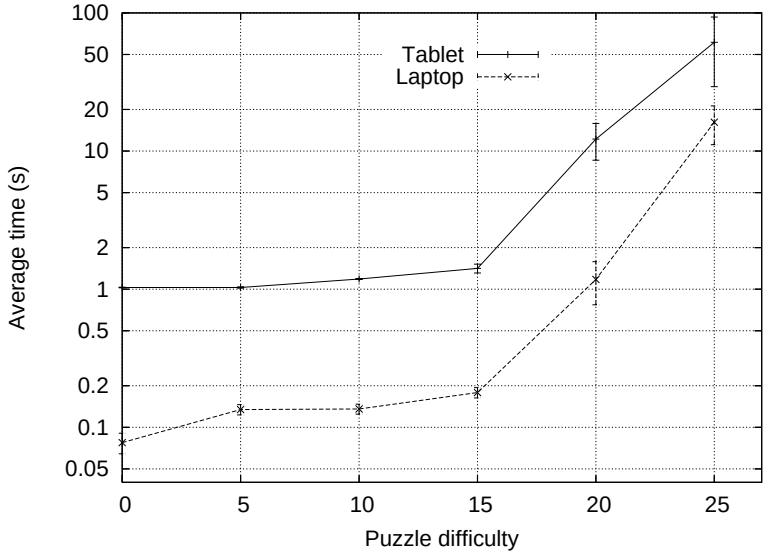


Figure 5.4: T2 processing time versus puzzle difficulty.

time needed to solve a puzzle by Tablet and Laptop. The tabulated results show a substantial increase of the standard deviation with the growing puzzle difficulty. As was noted by Nikander *et al.* [112] 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 set to ten.

There is a time limit during which the Initiator must find a solution to the cryptographic 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. Laptop, in con-

Table 5.1: 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

trast, would solve a similar puzzle 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.

Diffie-Hellman

The Diffie-Hellman (DH) key exchange protocol is used in HIP to exchange the DH public values of the hosts and produce a common 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 [98]. 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 practice, 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 5.5. 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 PC as the Responder. With the 768-bit DH Group, T2 processing time for Tablet is slightly higher and amounts to 234 ms, resulting in 340 ms on the average for the

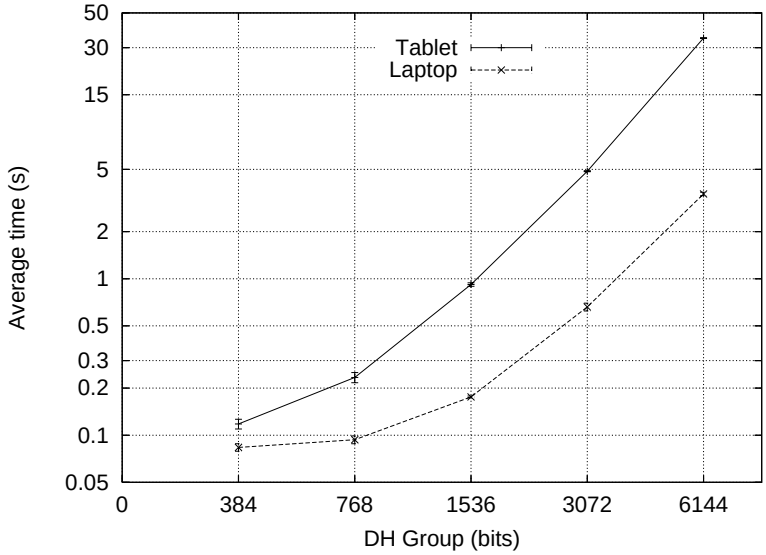


Figure 5.5: T2 processing time with different DH groups.

total duration of the HIP BEX with 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 Tablet as it results in the tremendous delays for the applications (over 5 and 35 seconds correspondingly). In comparison with Tablet, 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.

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 Nokia 770 Internet 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 the difference of the results in the processing time of the HIP control packets on Tablet (see, for example, Figure 5.4 and Figure 5.5).

Table 5.2: 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

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 utilized 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 Tablet, Laptop and PC acting as HIP hosts. The number of runs in each test was 100. Table 5.2 contains both median and average values of RTT for packets sent over plain IPv6 and over HIP. Here, the use of median values instead of mean values with standard deviations is more justified 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 association establishment. As an example, the tests IPv6(116B) and the IPv6/HIP in the Tablet-to-PC scenario contained one additional outlier caused by an unidentified reason. To describe the whole distribution, we use the cumulative distribution function (CDF) and show that the frequency of the outliers in our 100-number distribution is rare. Figure 5.6 presents the CDF for 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 that we measured in the PC-to-Tablet scenario by the ping6 tool in-

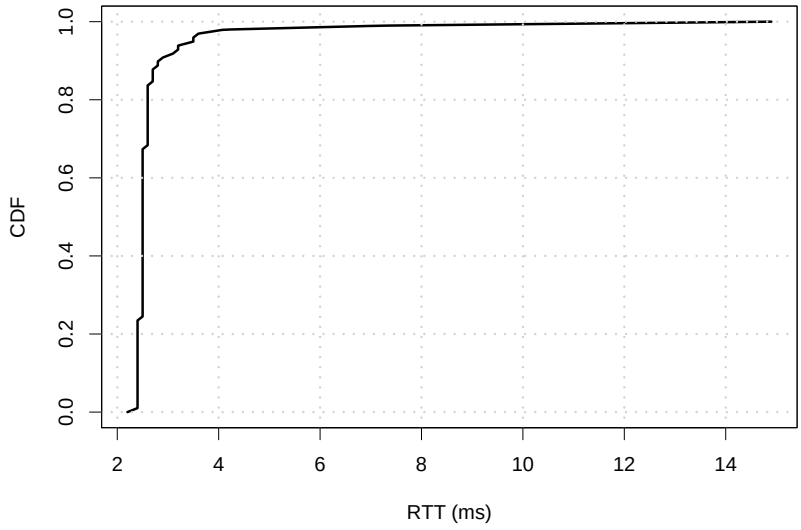


Figure 5.6: CDF for the RTT in the Tablet-to-PC scenario with IPsec.

cludes the transmission time of an ICMP ECHO_REQUEST message from PC to Tablet, processing time on both hosts and the latency of delivering an ICMP ECHO_RESPONSE back to 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 size of ESP headers to 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 message (IPv6 over HIP). The results indicate that increasing the size of the ICMP packets does not merely 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 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 af-

Table 5.3: 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	–

fects more seriously the lightweight devices than the ordinary desktop computers or laptops.

Throughput

IPsec ESP data encryption performed by Tablet can reduce the maximum achievable throughput over the wireless link. We measured TCP throughput by the iperf tool generating TCP packets to a correspondent node. It is necessary to 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 5.3. An average value of 4.86 Mbps represents an upper bound of the throughput achievable by 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 the theoretical maximum data rate of 54 Mbps, the Tablet's CPU or improperly implemented device driver imposed 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 Tablet, Laptop achieves 21.77 Mbps of the TCP data rate over

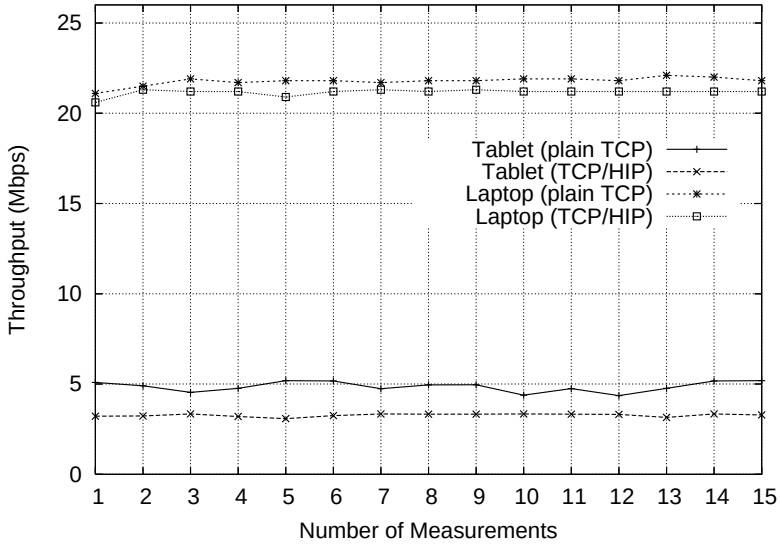


Figure 5.7: TCP throughput in an open wireless network.

the same wireless link (see Laptop-to-PC scenario). An interesting observation is that on Laptop the impact of the ESP encryption involved with HIP is small as compared to Tablet and equals 0.61 Mbps of the throughput decrease.

Figure 5.7 depicts the throughput results graphically and shows the distribution of the TCP and TCP/HIP throughput over a WPA-free wireless link. The graph illustrates the influence of HIP (ESP) on the TCP throughput as well as the difference in values achieved 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 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.

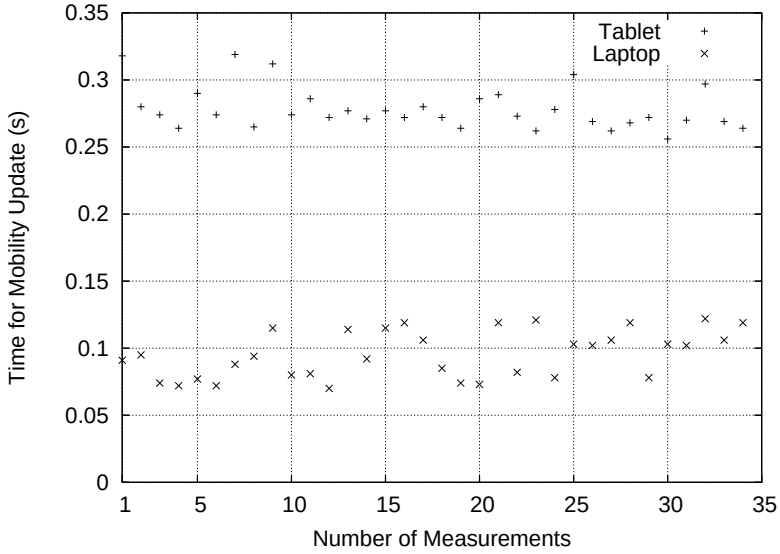


Figure 5.8: Duration of a HIP mobility update.

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 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 5.8). This time was necessary to exchange three HIP mobility control packets between Tablet and PC and it also includes the 1.5xRTT. Comparing to Tablet, the 1.6-GHz Laptop was capable of completing the three-way mobility update with the correspondent node (PC) in 100 ms on the average. One reason for a 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 reduced for applications. Once the correspondent node receives the first UPDATE packet it knows the Tablet's new location and can start transmitting 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.

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 applications requiring data transmission over WLAN. The objective of measuring battery lifetime on 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 5.4). 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 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 consumed 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 nature 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, 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.

Table 5.4: Power consumption by applications on Nokia 770.

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

5.1.3 Nokia 770 - summary of results

The above sections 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, 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's 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 [62] and MOBIKE [70], which rely on similar public-key and IPsec ESP operations.

5.2 Performance of HIP on Symbian OS

This section continues from Section 5.1 and reports on our experience with the Host Identity Protocol on another operating system, Symbian OS, and another type of devices, smartphones. While there are three open-source HIP implementations for various desktop operating systems, little experience is available with running HIP on mobile communications systems such as a cellular phone. In this section 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 section is based on the results published in our article *Performance of Host Identity Protocol on Symbian OS*[16] [66].

[16]© 2009 IEEE. Portions reprinted, with permission, from A. Khurri, D. Kuptsov, and A. Gurtov. 2009. Performance of Host Identity Protocol on Symbian OS. In: Proc. of the IEEE

To check whether running HIP on smartphones is feasible, we performed a set of measurements over WLAN with the Nokia E51 and Nokia N80 in different scenarios. Particularly, we measured 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 Symbian OS.

Symbian OS is one of the leading operating systems for smartphones. 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. In this section we evaluate applicability of the Host Identity Protocol 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 the section is structured as follows. Section 5.2.1 describes our ports of HIPL and OpenHIP implementations. In Section 5.2.2 we present our experimental network testbed followed by a description of scenarios and measurement tools in Section 5.2.3. Section 5.2.4 contains selected measurement results of the base protocol and their in-depth analysis. Section 5.2.5 concludes the section with a summary of key findings and conclusions.

5.2.1 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 setting up the development environment, examination of the existing HIPL and OpenHIP source code, preparation of the Symbian project structure and *makefiles*, compilation, debugging and testing.

International Conference on Communications 2009 (ICC'09), pages 1–6. IEEE.

Symbian OS networking architecture

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 [30].

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 [33]) 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 [30].

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.

Development environment

To start development for Symbian platform, we used 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 [33]. 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.

Project preparation

Before actual porting it was 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 [89] protocol (with BEET mode support) for communication with SADB.

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 the 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.

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 to compile code for the Symbian emulator. When the HIPL code was built for the target hardware with the *GCCE compiler*, the program behaved similarly with a Linux environment 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.

Limitations of 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 – the 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()* functions, multiple *I/O consoles* [32]. Because of Open C constraints our HIP ports for Symbian use only UDP sockets to send HIP control packets excluding a raw-socket alternative from 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 a 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.

Our HIP code ported to Symbian OS required *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 recompilation, could not be installed and used on any other S60 3rd edition mobile phone. To install the HIP daemon on another phone we needed to sign the package with its own IMEI number at *www.symbiansigned.com*. This made debugging on the target hardware inconvenient, required numerous rebuilding and resigning steps, and, at the same time, complicated the large-scale deployment of the software. It should be noted though that since the time of our project, Symbian Ltd. has been transformed to a non-profit organization, the Symbian Foundation, opening the Symbian platform code to developers, which among other things, should make porting to this platform easier and avoid many problems we experienced [4].

5.2.2 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

Table 5.5: 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

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 5.5. 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. Battery capacity in all phone models we used varies from 860 to 1050 mAh.

We measured 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 5.1 except that Nokia Internet Tablets were replaced with Symbian smartphones.

5.2.3 Measurement scenarios

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.

Table 5.6: 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

5.2.4 Performance evaluation

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

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 the *active* and *standby* phone state. We use terminology from [31] 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 a "screensaver").

As Table 5.6 indicates, the total average time for a 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 direction (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 5.6). Thus, time variation of the HIP base exchange performed in two opposite directions is insignificant. In our previous study [69] 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 [69] (presented in Section 5.1) indicates that the HIP base exchange performance on a Linux-based Nokia 770 Internet Tablet is better than on a 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 5.6 indicates that the OpenHIP implementation shows slightly better performance than HIPL, establishing a HIP association between E51 in the active state and the server on the average during 3.09 seconds and in the opposite direction in 2.76 seconds. Still, the effect of the standby state in the OpenHIP case is less significant than with HIPL (OpenHIP 35% against HIPL 47% time decrease when switching to the 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 between two mobile phones. Our preliminary tests show (see Table 5.6) that two Nokia E51 smartphones in the 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 the mobile phone user to interact with the handheld (thus, activating its display) while using applications that might benefit from HIP.

Table 5.7: 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

Asymmetric key pair creation

Table 5.7 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 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 need 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 key management on mobile phones. One approach can be to generate the keys on a PC and securely transfer them to the phone. This can be performed by the phone 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 during its manufacturing. In this case, any additional user actions are avoided. Further details on the key distribution on mobile devices are left outside of this work.

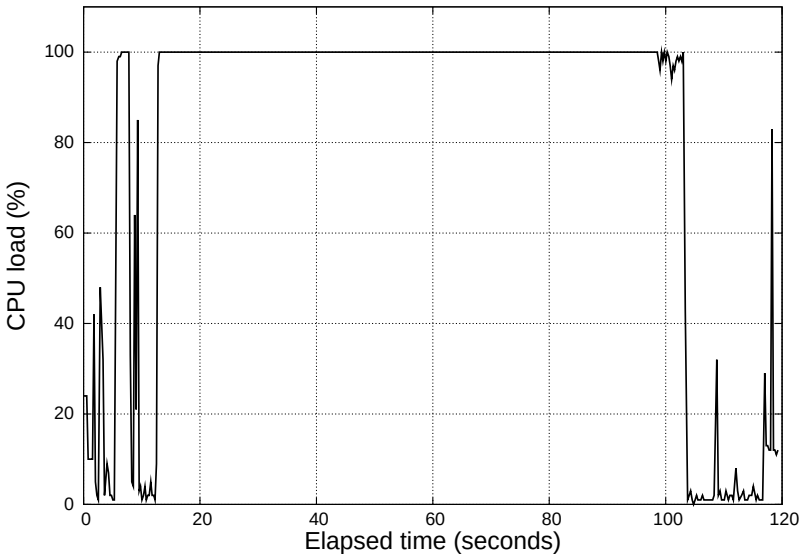


Figure 5.9: HIPL daemon initialization. CPU load on E51.

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 creation on E51 is presented in Figure 5.9. Most of the 2-minute measurement period the CPU load stays at 100% and this corresponds to the daemon initialization (up to the 12^{th}-second timestamp) and the generation of four different public-private key pairs (the interval from 12^{th} to the 104^{th} second). The rest of the graph (from the 104^{th} 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.

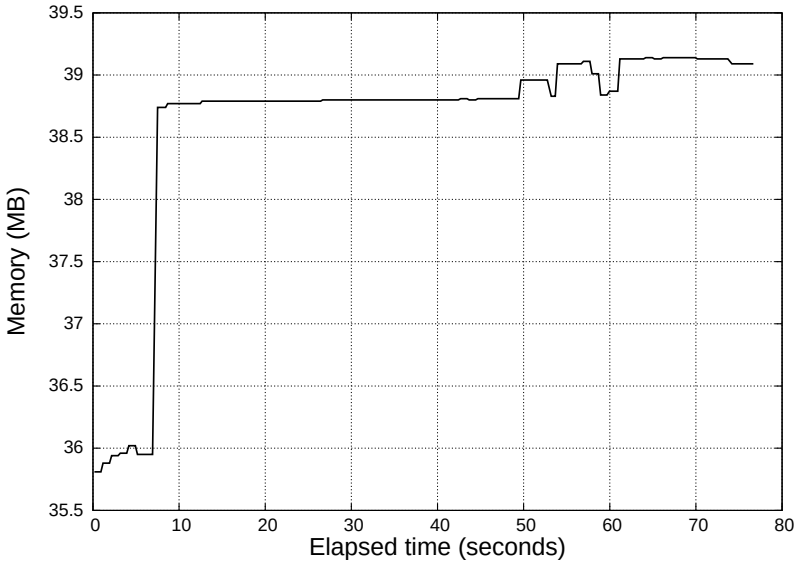


Figure 5.10: OpenHIP daemon initialization with BEX. RAM usage on E51.

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 exclusive use by the system. This reduces the number of user applications that can be simultaneously ran on the device. According to our profiling data, the memory usage 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 5.10 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 8^{th} to the 50^{th} 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 50^{th} to the 80^{th} second), the memory usage does not grow drastically but only increases by half a MB during 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.

Power consumption

Figure 5.11 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-minute measurement period. The peaks on the graph between the 8^{th} second and the 53^{nd} second timestamps show the maximum power consumption over the measurement period and they account for creation of two DSA and two RSA key pairs.

To understand 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 a HIP 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.

5.2.5 Nokia E51 - summary of results

This section 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 recommen-

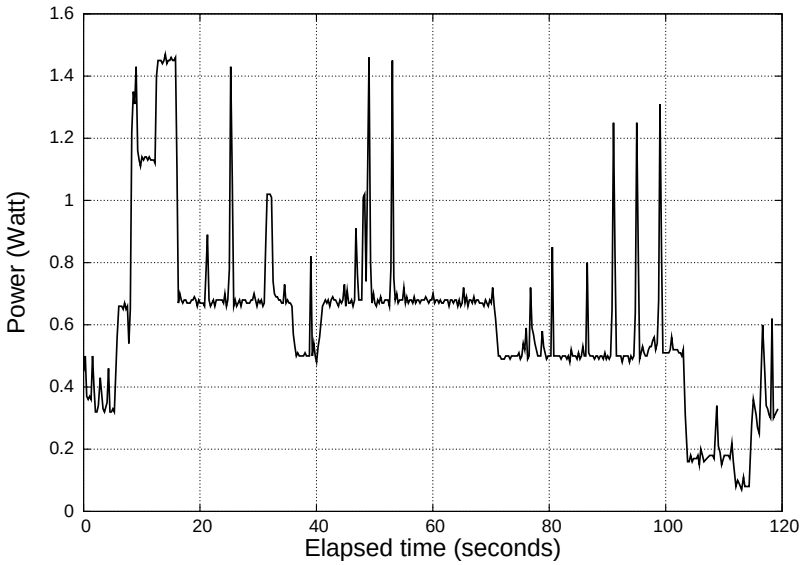


Figure 5.11: HIPL daemon initialization. Power consumption on E51.

dations on the use of public-key cryptography on lightweight Symbian OS mobile phones.

- A single HIP base exchange between the Nokia E51 and a 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 delivered to user or generating the keys on a PC and transferring them to the mobile phone.
- Key creation and signature operations boost CPU utilization and consume 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.

6 On Application of Host Identity Protocol 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*[17] [78].

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. Preliminary experimentations with the presented architecture have been done in panOULU [3], a public city WLAN in Finland.

[17] © 2009 IEEE. Potions reprinted, with permission, from 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), pages 1–9. IEEE.

The rest of this chapter is structured as follows. Section 6.1 discusses the issues with existing wireless networks and states the objectives we aim to achieve with our architecture. Section 6.2 presents our distributed WLAN authentication architecture. In Section 6.3 we describe our port of the HIPL implementation to OpenWrt platform and experimental testbed used for our measurements. Section 6.4 contains the measurement results of CPU and memory utilization on two different AR models. Section 6.5 concludes the chapter.

6.1 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 investigation in case of network misuse.

Open wireless networks (such as presented in Figure 6.1) 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 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 [18] and 802.1X [19], necessitate to look for 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., [74, 12]). However, none of the methods achieves all

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

[19] 802.1X vulnerabilities, `http://www.microsoft.com/technet/community/columns/secmgmt/sm0805.mspx`

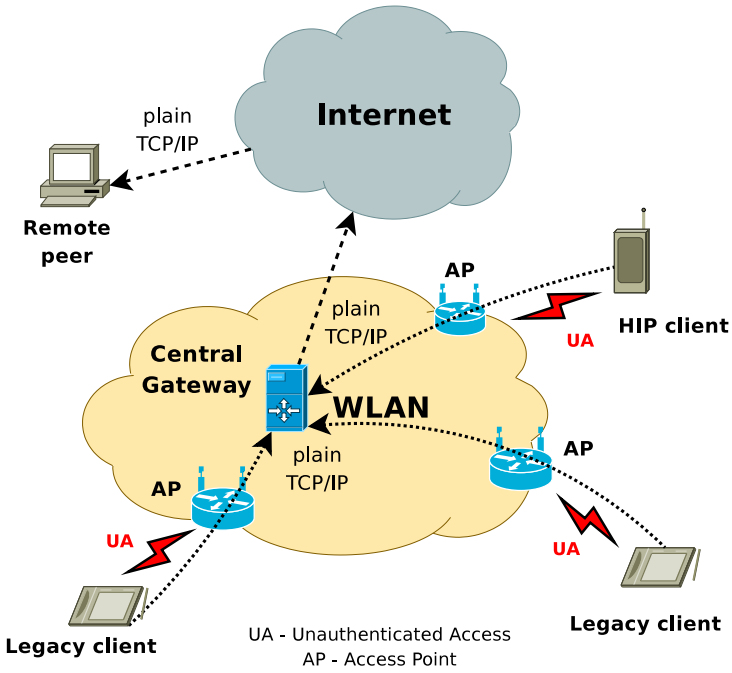


Figure 6.1: Open network access model.

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.

6.2 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.

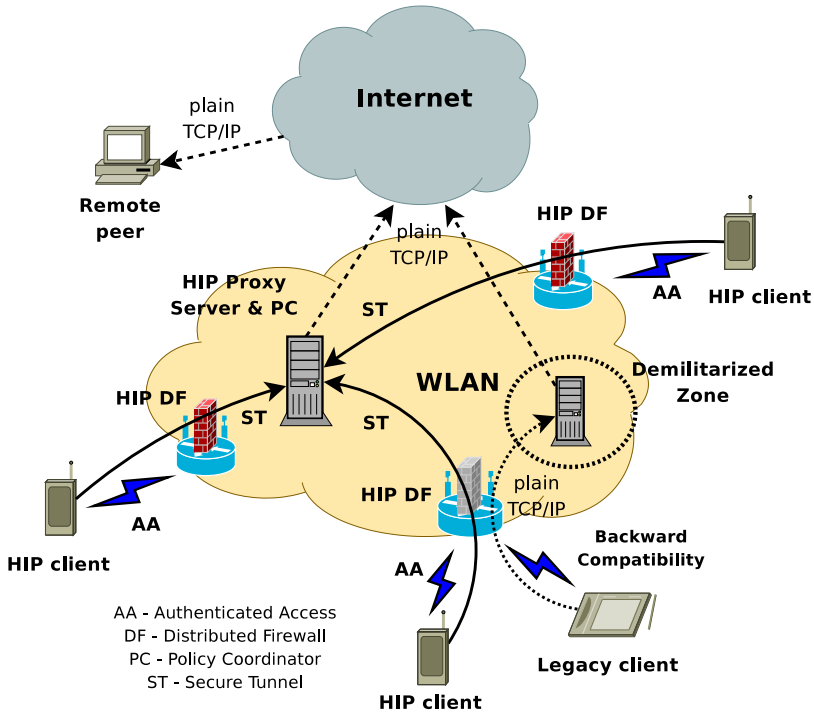


Figure 6.2: Distributed authentication model.

6.2.1 Architectural components and principles

The general view of our architecture is shown in Figure 6.2. 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 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 a *distributed firewall* (see Figure 6.2). 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 [47]).

Checking 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.* [46]. 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.

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

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

- 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) 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 6.2 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.

6.2.3 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 the one 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 [9, 40, 57, 138].

6.2.4 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 [77] 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.

6.2.5 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 6.4.2 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 6.2. 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).

6.3 Experimental testbed

This section presents our experimental testbed. First we describe the HIPL [1] porting process and highlight the challenges we confronted during migration to the OpenWrt platform. Next, we show the components of our network setup and introduce the status of preliminary architecture tests in the panOULU WLAN.

6.3.1 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 an 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 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.

6.3.2 Experimental setup

Our network setup (see Figure 6.2) 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, both HIP-aware and non-HIP. The clients used two publicly available HIP implementations, HIPL and OpenHIP [116]. All components of our experimental testbed for distributed user authentication in a wireless network are illustrated in Figure 6.2.

6.3.3 Considerations for deployment

Our system works in a way that 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 user 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, e.g., panOULU, our architecture can be easily extended to provide backward compatibility for legacy clients. This can be realized by a simple port switching technique on the Wi-Fi ARs as described in Section 6.2.5.

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.

6.3.4 Experiments in panOULU

We have conducted preliminary experiments with certain components of our authentication architecture in panOULU, a city-wide WLAN in Finland, in 2008. We have installed an HTTP/HIP proxy on the main network gateway. The proxy allowed 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 initial 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 the network with the running HIP firewall and proxy extensions. First experiments indicated that this low-cost AR model is stressed by the HIP proxy component that, in turn, is not able to serve multiple clients simultaneously.

6.4 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 [69, 66], which studied HIP performance on the Linux-based Nokia Internet Tablets and Symbian smartphones, served as a good reference. In addition, HIP has been evaluated on stationary Internet hosts with conventional desktop-like resources [51, 49, 60, 117].

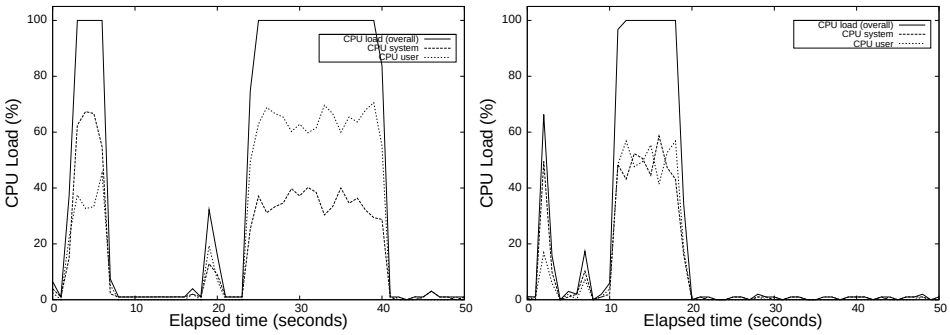
We have made an interesting observation that 100% CPU utilization does not necessarily indicate a performance issue for a particular mobile device. Rather, this can be interpreted as the utilization of all available resources by running applications when the system allocates maximum capacity to them. 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.

6.4.1 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 of the architecture.

Figures 6.3a and 6.3b illustrate a copying task of a 30-MB file over SSH and HIP, while the traffic was 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 6.3a; the time interval from the first to the fourth second in Figure 6.3b) corresponds to the HIP base exchange between



(a) Fonera (b) Avila

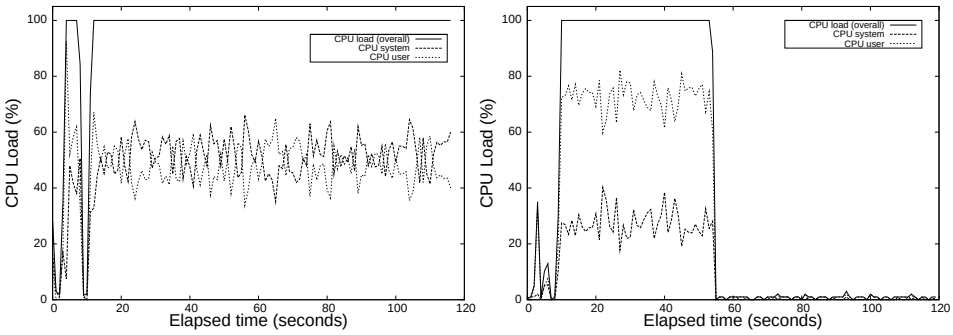
Figure 6.3: CPU load in the firewall mode.

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 19^{th} second in Figure 6.3a; the seventh second in Figure 6.3b) accounts for the SSH key exchange between the mobile client and a remote peer. Finally, the interval from the 23^{rd} to the 42^{nd} second in Figure 6.3a and from the 10^{th} to the 20^{th} second in Figure 6.3b 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.

6.4.2 Proxy mode

This section presents the results obtained on Fonera and Avila running in the HIP proxy mode. Figures 6.4a and 6.4b depict the CPU utilization during the bulk



(a) Fonera (b) Avila

Figure 6.4: 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 6.4a), while on Avila the same operations result in less than 40% of the CPU utilization (the first peak in Figure 6.4b). 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 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.

6.4.3 Mode selection

The analysis of the measurement results allows us to give the following recommendations on deploying the presented architecture:

- For the areas with a small rate of connections, it is sufficient to have low cost devices such as La Fonera FON2100 to authenticate WLAN 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 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.

6.5 Summary of results

In this chapter, we have proposed a HIP-based distributed authentication architecture that can offer means to solve security and mobility issues and ensure authentica-

tion of mobile users to wireless LANs. 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 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.

7 Applying Host Identity Protocol in Wireless Sensor Networks

This chapter is primarily based on our previous work *On Application of Host Identity Protocol in Wireless Sensor Networks*[20] [67].

With proliferation of advanced wireless sensor platforms featuring multiple connectivity options, input/output interfaces, scalable processing units and large memory storage, the popularity of wireless sensor networks (WSNs) grows opening up space for new applications for civil and military use. Security concerns of WSNs emerge, too, and despite active research in this area, a number of issues remain open.

A variety of well known attacks in WSNs aiming to violate confidentiality and integrity of sensitive data, to replicate node identity and to cause a denial of service, call for reliable and efficient security measures. Numerous research initiatives have investigated the question how different means of cryptography can be exploited in communications of wireless sensor nodes. Public-key cryptography (PKC) well known for its advantages over, e.g., symmetric algorithms in terms of better key management and scalability has been for a long time thought of as an infeasible cryptography class due to its computational complexity for low-power sensor motes (e.g., [85]). However, substantial research that followed has disproved earlier claims and shown that PKC and especially Elliptic-Curve Cryptography (ECC) are sensible even for weak sensor devices [84, 121, 126].

Although advanced sensor platforms such as *Imote2* [2] have further diminished the issue of computational complexity of cryptographic operations, energy consumption of on-board computations and wireless communications in WSNs is still a big concern. In addition, as related work shows, despite availability of individual security mechanisms applying them standalone is in most cases insufficient as it leaves possibilities for attacks [84, 137]. Hence, for better security a WSN often requires a combination of different components that together can be used in building of adaptable security protocols suitable in distinct circumstances.

In this chapter, we report our experience of running existing IP security components on wireless sensor motes and present a set of corresponding performance measure-

[20] © 2010 IEEE. Portions reprinted, with permission, from A. Khurri, D. Kuptsov, and A. Gurtov. 2010. On Application of Host Identity Protocol in Wireless Sensor Networks. In: Proc. of the 7th IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS'10), pages 358–365. IEEE.

ment results. We propose to use the Host Identity Protocol (HIP) [97] as the main component for building WSN security. In WSNs, HIP can use public keys for identification of sensor nodes, PKC signatures for their authentication, a Diffie-Hellman (DH) key exchange to generate a common session key for a pair of nodes, a puzzle mechanism for protection against DoS attacks in certain network parts and the IPsec protocol for optional encryption of application data. By integrating several mechanisms HIP can solve many security problems of WSNs.

In the following sections we first discuss how HIP can strengthen security of WSNs, then describe our pilot environment consisting of Imote2 sensor platforms [2] and finally present our measurement results and analysis of HIP computational and energy costs for this type of devices. The rest of the chapter is organized as follows. In Section 7.1 we explain why we selected the Imote2 sensor platform as the target hardware for our experiments. Section 7.2 shows how HIP can be applied in WSNs. Section 7.3 contains performance evaluation of the protocol components on the Imote2 sensor platform and section 7.4 concludes the chapter.

7.1 Scenarios with Imote2

The choice of Imote2 as the target hardware in our work was a result of the following observations. On the one hand, most of the research projects in the WSN field to date concentrated on extremely constrained TinyOS-based sensor motes with 8-bit processors (clocked at few MHz) and kilobytes of RAM. Unlike previous studies, we focused on the Imote2 sensor platform that, due to its advanced features, enables novel resource-demanding applications and represents a new generation of "smart" sensors. Support of Linux OS adds potential value of porting existing protocols, libraries and applications. Platforms such as Imote2 provide high potential for medical, industrial and environmental monitoring applications [6, 124]. One of the Imote2's advantages over less powerful sensors is the ability to decrease the WSN overall energy and bandwidth consumption by processing as much data on board and transmitting as little data over radio as possible [99]. All this makes Imote2 a prospective candidate for future WSN deployments in a number of areas.

On the other hand, having a combination of scalable CPU, a MMX coprocessor, considerable RAM and storage space, small device form-factor, multiple I/O interfaces and connectivity options [2], Imote2 can become an essential architectural component of a multi-layered security framework for different WSNs. It can serve, e.g., as a sink or an aggregator of sensed data within a cluster of less capable nodes,

performing data processing and secure forwarding to a remote site by employing strong cryptography. It can also act as an edge router in a medical patient area network (PAN), forming a secure gateway between the PAN and a remote service (e.g., a doctor). WSNs composed exclusively of powerful Imote2s can form reliable backbone networks.

7.2 How a WSN can benefit from HIP

In this section we discuss what benefits HIP in its "standard" or existing form can provide for WSNs and how certain components of the protocol would apply to such networks and devices. Considering the scenarios described above as primary ones for prospective use of Imote2, we assume that HIP can be used between an Imote2 and a remote server or another communication device, between two Imote2 nodes (e.g., being cluster heads), and, in certain cases, between an Imote2 and more constrained sensor nodes (the feasibility of running PKC primitives on tiny sensor nodes has also been shown previously [84, 121, 126]).

7.2.1 Public key certificates

Most of the emerging commercial WSN applications will be used by closed groups with a set of different rights, thus requiring appropriate access control mechanisms. Each member of a WSN needs to prove its identity to all other members, which can be done using certificates. In case of HIP, the Host Identity (or the public key) of a node has to be signed by a certificate authority (CA) all other nodes trust. All nodes are expected to possess the public key of the CA and, thus, be able to verify and trust in authenticity of information signed with the CA's private key. In addition to a CA, the public key infrastructure (PKI) of WSNs may include a repository that stores the list of untrusted/revoked public keys or identities. The selection of PKI components depends upon desired application requirements. For instance, in case of a medical WSN, we assume the presence of a central CA. If a patient's node does not trust in the certificate it received in a HIP BEX from a doctor, it may want to contact the CA for checking the state of the certificate in a certificate revocation list (CRL). In this way the sensor node can verify the validity of the doctor's certificate and check whether it was actually revoked before the expiration date or upon an event of compromising or stealing the device holding the public key [39]). Communication with the CA has to be itself secure, which can be achieved by establishing a separate HIP association between the sensor node

(the patient in this case) and the CA (provided the CA is the entity trusted by each participant).

In case of an emergency situation, when the patient's node in the above example cannot contact the CA prior to communication with the doctor due to time limits, insufficient network coverage, the CA being offline or other constraints, the doctor (while approaching the patient) can request from the CA a token, which it can include into the BEX with the patient. The token would be a temporary certificate valid for a limited time period (as needed by the case), which will prove that the doctor's certificate was not revoked.

PKC certificates (e.g., X.509) can be added to HIP signalling traffic as described by Heer and Varjonen [48], enabling identity (public key) authorization and verification of the certificate status. However, to build a complete PKI, a sensor node will need additional signalling with a server storing CRLs. This will require protocol modifications, further details of which we leave out of the thesis scope. In case of a fully autonomous remote WSN with no central CA, CRLs have to be stored in a distributed manner (see, e.g., [35]). Additional cooperative security mechanisms should be applied in such scenarios to check the status of certificates and perform revocation of malicious or malfunctional nodes (see, e.g., [76]).

7.2.2 Authentication of sensor nodes

With the proven identity by means of PKC certificates, a sensor node will be able to authenticate itself to other WSN nodes, which is needed already during the node admission phase. HIP uses PKC signatures for authentication of communicating parties. To improve efficiency of these operations and preserve energy for sensor motes, "traditional" RSA (Rivest, Shamir and Adleman) or DSA (Digital Signature Algorithm) algorithms can be replaced with Elliptic Curve Cryptography (ECC) algorithms. We present performance of RSA and DSA operations on the Imote2 sensor platform in Section 7.3.

7.2.3 Role management

Applications of WSNs may require separation of roles to cluster nodes in the network or to delegate specific rights to certain sensor motes. For instance, there can be several intermediate sink nodes in a WSN that would have more computational and battery resources or a special hardware module to store sensitive information

securely. Depending on a particular architecture, the delegation of rights and the separation of roles can be added to HIP again in a form of PKC certificates containing bindings between a sensor node identity and a subset of its roles and rights in this WSN.

7.2.4 Key exchange and secure channel establishment

To securely exchange sensed information between each other and with a sink or a central entity, sensor nodes should be able to establish a secure communication channel. For this purpose, HIP utilizes a DH key exchange protocol that generates a shared secret key for two nodes over an insecure channel. As will be seen in Section 7.3, a DH key exchange makes a major impact on duration of a HIP base exchange and, hence, it is wise to replace it with a much more efficient ECDH (Elliptic Curve Diffie-Hellman) exchange for constrained wireless sensor nodes.

The shared keying material resulted from a key exchange serves as an input to creation of IPsec ESP security associations used for optional network-layer encryption of sensitive sensor application data. The applicability of IPsec in WSNs, however, has to be considered carefully as it adds extra bytes to the protocol header and increases energy cost per byte (we evaluated the impact of IPsec encryption on data throughput and energy consumption in our previous work [69]). Given the limited frame size of the $IEEE$ 802.15.4 radio commonly used in WSNs (102-bytes on the medium access control layer [95]), as well as battery constraints of sensor nodes, we assume that the use of IPsec in WSNs will be limited, though not completely infeasible. One option is to utilize the keying material derived from a DH exchange for the link-layer AES-based encryption implemented in hardware on most $IEEE$ 802.15.4 chips. Another option is to use the IPsec ESP mode with the AES CCM mode as suggested by Housley [53].

7.2.5 Protection against attacks

HIP contains a number of mechanisms that make it resistant to different attacks. However, certain protocol techniques provide no direct benefits to WSNs. For instance, the puzzle mechanism, that protects HIP responder from potential DoS attacks and allows to postpone creation of the state until receiving a valid I2 packet [98], would likely not be able to prevent a battery-exhausting DDoS (Distributed Denial-of-Service) attack, given the vast difference in resources of the potential attacker (e.g., a network of zombies) and the victim (i.e., a lightweight sensor

node). To eliminate DoS and DDoS attacks, we suggest to use additional access control mechanisms (e.g., a HIT-based firewall) that would allow to filter out unwanted or unregistered clients already at an early stage prior to processing of HIP control packets. Limited number of legitimate clients is a fair assumption for the above-mentioned scenarios where an Imote2 acts as a cluster head or a gateway for less capable nodes. HIP can be used in such scenarios to authenticate the Imote2 and a remote host (an "operator", a sink etc.) to each other and to set up a secure communication tunnel between them. However, provided "non-public" nature of the network, the possibility of an external DoS attack would be rather small and, hence, the puzzle mechanism is of no use in this or similar scenarios.

Please note that we only imply attacks on the network layer that are a result of retransmissions or intentional flood of spoofed HIP packets. We do not consider other possible types of DoS attacks such as, e.g., jamming radio channel with a signal of interfering frequency. HIP packet protection against replay attacks is described thoroughly in the protocol specification [98], please refer for details to it. Sybil attacks (or taking on multiple identities) are not feasible in HIP scenarios as long as HITs (public keys) of communicating hosts are signed with the private key of a CA, which should be a prerequisite as stated earlier. Among other methods, the use of a tamper-proof device for storing private keys of WSN nodes can prevent attempts of their "cloning". In general, we consider the "standard" HIP to be useful in various WSN scenarios but particularly in those presented in the beginning of this section.

7.3 Performance evaluation

While HIP is a mature and standardized protocol for end-to-end security, it can adjust its parameters to different values, providing less or more security and, at the same time, communications and processing overhead to a communication system. To evaluate suitability of HIP in its "original" or "standard" form for wireless sensor networks, we performed a number of measurements on the Imote2 sensor platform. The results of these measurements are presented in this section and especially energy consumption of particular protocol operations (including possible lightweight alternatives that are not part of the standardized HIP) provide a good basis for analysing feasibility of IP security in WSNs and building an adaptable security framework for lightweight communication devices.

We first introduce technical specifications of the Imote2 sensor platform, then describe the methodology for our experiments and finally present measurement results

including HIP computational and energy cost.

7.3.1 Sensor hardware specifications

Technical specifications of tested Imote2 sensor motes can be found from a product datasheet [2]. They are also summarized in Table 7.1. Comparing to other well known sensor motes (such as, e.g., TelosB, IRIS, MICAZ), Intel Mote 2 is an advanced platform featuring an Intel XScale processor and 32 MB of RAM and enabling computation-intensive wireless sensor applications. Dynamic voltage scaling allows to scale CPU frequency from 13 to 416 MHz and to choose the most appropriate power mode for each application. Different sleep modes of the processor allow to save energy when applications are inactive [2].

Imote2 sensor nodes are usually shipped with .NET Micro Framework preinstalled but can also be flashed with TinyOS and Linux operating systems. While considerable research has been carried out to evaluate security on wireless sensor nodes running TinyOS, we concentrated our study on Imote2 sensors powered by a Linux 2.6.14 kernel and with support of *cc2420* and *ieee802154* radio modules.

7.3.2 Measurement results and analysis

We present only a subset of the most important measurement results including computational complexity of individual cryptographic components and their energy cost for an Imote2 sensor node.

Computational complexity of HIP

To see how the total duration of the HIP base exchange is affected by its individual components we have performed a set of isolated measurements. The results allow to see which elements make the most critical impact in terms of computational complexity and energy cost to an Imote2 sensor platform. The components include asymmetric key generation, signature operations (RSA and DSA versus ECDSA), Diffie-Hellman key exchange (DH versus ECDH) and the puzzle mechanism. The results of the measurements not presented separately are collected in Table 7.4.

Asymmetric key pair generation Table 7.2 contains the median values of the time needed to generate a public-private key pair of different size on an Imote2 sensor.

Table 7.1: Technical specifications of Imote2 sensor platform [2].

	Intel Mote 2	
CPU	PXA271 XScale	13 – 416 MHz
Memory	SRAM	256 KB
	SDRAM	32 MB
	Flash	32 MB
Radio	Transceiver	TI CC2420
	Frequency	2.4 GHz
	Data rate	250 kbps
	Range	∼30 m
Current draw (at 13 MHz)	Deep sleep mode	390 μA
	Active, radio off	31 mA
	Active, radio on	44 mA

Each median value has been calculated over a corresponding 100-hundred sample of time values. The results indicate an expected exponential growth of the key pair generation time with increasing the key length. With conventional 1024-bits keys, the median time to generate an asymmetric DSA key pair on Imote2 equals to 12.1 seconds. The generation of an equivalent RSA key pair is much faster and can be accomplished within 2.3 seconds. It is worth pointing out that generation of public-private keys with the length over 2048 bits on Imote2 would produce a delay of seven minutes with DSA and almost one minute with RSA.

As will be seen in the next section, long generation times of an asymmetric key pair on Imote2 will result in a high amount of consumed energy. On a condition that public-private keys on sensors are normally generated infrequently (upon deployment into a network or when the old key is revoked), this computational and energy cost should not be serious for an Imote2 node.

Long generation delay of a public-private key pair on Imote2 sensors illustrates the necessity to look for different approaches for managing keys in wireless sensors net-

Table 7.2: Creation of a key pair of different size on Imote2.

	Median time (sec)			
↓ Algorithm / Key (bits) →	**512**	**1024**	**2048**	**3072**
DSA	2.1	12.1	87.5	280.9
RSA	0.5	2.3	17.2	45.4

works, especially in those with nodes being more resource-constrained than Imote2. One approach can be to generate keys on a more powerful node and securely transfer them to each sensor in the network via a sink node. This, however, might be a challenge when the need for rekeying appears during a long absence of the sink node. Also, communication between the sink and the rest of nodes has to be secure. Further details about key distribution in wireless sensor networks are left outside this work as a topic that in itself is a wide and extensively studied research area.

Signatures and verifications Table 7.3 shows computational complexity of operations on public-key signatures. It contains average times needed to generate and verify a signature on an Imote2 sensor platform using RSA and DSA keys of different length. The numbers are the results of benchmarking using the OpenSSL speed utility on an Imote2. The OpenSSL speed is a benchmarking suite that measures how many cryptographic operations can be performed in a system during particular time. A comparison of "traditional" asymmetric algorithms such as RSA and DSA with Elliptic Curve Cryptography (ECC) signatures on Imote2 will be presented in Section 8.2.2.

DH key exchange Diffie-Hellman (DH) key exchange is used to establish a shared secret key between two communicating parties. In HIP base exchange, it accounts for a major delay. BEX responder computes its own DH public value A using a secret key and sends A to the initiator in an R1 packet in clear. The initiator then calculates its own DH public value B using its own secret key and generates a shared session key K using the responder's public value A it received earlier. The initiator keeps the key K in secret, while shares its public value B in an I2 packet with the responder. The responder then uses B to generate the same secret key K, as the

Table 7.3: Duration of signature operations on Imote2.

	Average time (ms)			
Algorithm →	**RSA**		**DSA**	
↓ Key size	*sign*	*verify*	*sign*	*verify*
512 bits	4.1	0.3	4.2	3.8
1024 bits	20.0	1.1	11.2	12.0
2048 bits	127.1	3.9	38.2	43.9

initiator already has [98, 20]. In total, a DH exchange includes two calculations of a DH public value (DH compute) and two generations of a shared secret key (DH shared), performed on two hosts. Each peer, thus, does half of this job, i.e. one *DH compute* and one *DH shared* operation.

Our measurement results indicate that with a default 1536-bit DH group it takes 349.5 ms for an Imote2 sensor node to create a DH public value and 348.4 ms to generate a shared secret key, resulting in an overall 697.9 ms-delay for a complete DH exchange per sensor. In contrast with the 1536-bit DH group, the use of a weaker 768-bit DH group decreases the generation time of a DH secret to 84.1 ms. The above numbers are the median values calculated over 100-hundred samples of corresponding durations of DH operations measured by the benchmarking utility from the HIPL project source code [1]. The benchmarking utility runs pieces of the protocol code that implement respective functions/operations.

Processing of puzzle The puzzle mechanism in HIP intends to decrease the possibility of DoS attacks and allows to postpone creation of the state at the responder of a HIP BEX until receiving a valid I2 packet. The idea is to force the initiator to spend certain amount of CPU cycles for solving a cryptographic puzzle and to let the responder postpone creation of state until reception of an $I2$ message [98]. Figure 7.1 illustrates the dependency between puzzle difficulty and puzzle processing (solving) time measured on an Imote2 sensor node. The time spent by an Imote2 to solve a cryptographic puzzle rises exponentially from 20.1 ms with puzzle difficulty $K = 0$ to 143.6 seconds with $K = 25$. It is interesting to see that as in the

case with a Nokia Internet Tablet (Figure 5.4), puzzle processing time starts rising dramatically when the puzzle difficulty is set over 10. In turn, the efforts put by the initiator on solving a "zero" and a 10-bit puzzle are merely equal, thus, in our opinion, it makes sense to use this mechanism starting with the difficulty $K = 15$.

The numbers used to plot the graph are the median values of samples of puzzle processing times benchmarked on an Imote2 sensor mote. For $K = \{0, 5, 10, 15\}$ the corresponding sample size was 500 values, whereas for $K = 20$ and 25 the samples consisted of 100 values. The benchmarking results of puzzle processing time have been also verified during a real base exchange between two sensors. In general, solving a puzzle in our experiments, as well as in related work [112], has shown to be an "indeterministic" operation with notably varying processing times, especially with high values of K.

HIP specification suggests that the choice of puzzle difficulty by the responder should be based on its degree of trust with the initiator, present load or other factors [98]. Among "other factors" we see specifics of communicating hosts. For instance, we find it logical to avoid setting the puzzle difficulty to high values if the initiator is a lightweight client (a resource-constrained device) bearing in mind its limited ability to produce a true DoS attack. Otherwise, processing of difficult puzzles on a weak sensor device would notably raise energy consumption and shorten the device lifetime. Hence, when communicating with such a resource-constrained initiator, the responder may initially use a low puzzle difficulty but reset it to a high value upon detecting that it is under an attack. Adjusting the difficulty dynamically depending on multiple factors should be the goal while using such client puzzle mechanisms [98, 8].

In an opposite case, when the initiator is a powerful device (a server-like computer) and the responder is a sensor device, the applicability of HIP puzzle mechanism to protect the responder from a DoS attack is questionable. This is because the initiator having merely unlimited computational resources may quickly solve cryptographic puzzles it receives and flood the lightweight responder with valid $I2$ packets. A slight chance to eliminate a DoS attack can be to set the difficulty to a maximum possible level. For instance, our experiments show that for a 3GHz-Intel Pentium 4 computer solving a 25-bit puzzle takes from one second to several tens of seconds. The same computer is able to solve a 28-bit puzzle (the maximum difficulty available in the HIPL implementation) as fast as in 5 seconds, although in some our tests this operation took 8 minutes, which definitely exceeds all expectations about a sensible lifetime of a puzzle and acceptability of such a delay by an application/user. Of course, there can be attackers with supercomputing capabilities able to solve even

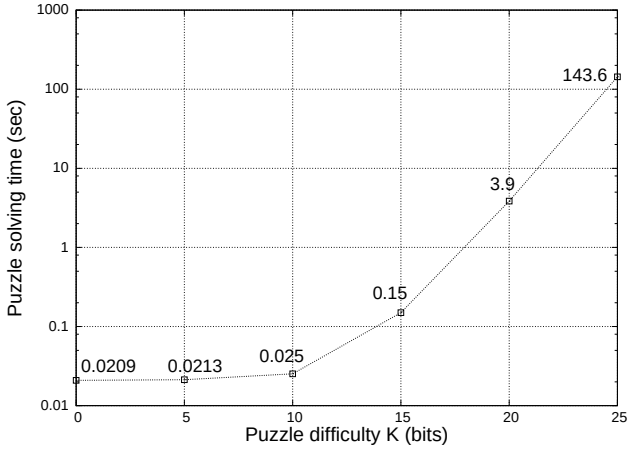


Figure 7.1: Puzzle solving time on Imote2

most difficult puzzles in a relatively short time. Though presence of such an attacker near a remote wireless sensor network is doubtful.

As a summary, the HIP puzzle mechanism can be used as a means of protection against DoS attacks with HIP in wireless sensor networks but it has to be used with care. It is likely not applicable in scenarios where the attacker is a powerful device. As said earlier in this chapter, the mechanism would likely not be able to prevent a battery-exhausting DDoS (Distributed Denial-of-Service) attack either, given the vast difference in resources of the attacker (e.g., a network of zombies) and the victim (i.e., a lightweight sensor node). The right choice of parameters such as puzzle difficulty depends on multiple factors including but not limited to the level of trust between communicating parties, the load of peers at present moment, the lifetime of a puzzle set in the implementation and the type of peer hardware. Detecting the "real" hardware type of your peer is, in turn, challenging if not impossible, especially prior the moment a secure communication context is set up between the hosts.

BEX total duration The total duration of a HIP base exchange in our tests took 1.1 seconds in the scenario with an Imote2 and a server (connected via a USB network with the average RTT=3.7 ms), and 1.7 seconds in the case with two Imote2 sensors communicating to each other over IEEE 802.15.4 radio with RTT varying from 13 to 127 ms on the average for packets ranging from 64 to 1028 bytes in size.

The above BEX numbers are median values of the corresponding samples of measured BEX duration. In a real wireless sensor network, this delay for an association establishment should not become a concern unless such exchanges are frequent. Instead, a total energy cost of a HIP communication produces more considerations, as it will depend on security requirements and threats and vary from scenario to scenario. This cost will be accumulated from the energy amount consumed by all operations constituting the security protocol in use. In the next section, we present our measurements of energy consumption of isolated cryptographic operations and other protocol components.

One of the factors affecting the total BEX time in a wireless sensor network can be the frame size on the link layer (media access control (MAC) layer). In our experiments, sensors communicated with each other over IEEE 802.15.4 radio with the maximum frame size on the MAC layer equal to 102-bytes [95]. A single HIP exchange contains four packets exchanged between the initiator and the responder. The approximate size of these packets varies from 44 to 628 bytes. Obviously, the IP packets with size over 102 bytes need to be fragmented to be transmitted over the IEEE 802.15.4 media. Fragmentation imposes its own overhead not only by fragmenting each large IP packet and adding to each frame an IP header but also by taking more time to transmit all fragments to the destination (and receiving acknowledgements). During our tests we observed an effect of fragmentation and lost fragments when a HIP base exchange could not complete without retransmissions for a long time. Unfortunately, we did not use a sniffer between the communicating sensors nor we could utilize *Wireshark* to see what actually happens on the link layer in our system (e.g., to detect the lost frames etc.). However, the duration of a "successful" HIP BEX was affected by IP fragmentation, as the $R1$, $I2$ and $R2$ packets were of size 612, 628 and 220 bytes respectively. Numerically, an approximate impact of the fragmentation on the duration of a HIP base exchange can be assessed by analysing transmission of ICMP packets in the same network. In our experiments, RTT varied from 13 to 127 ms on the average for the packets ranging from 64 to 1028 bytes. Thus, the fragmentation produced an additional overhead in the range of 100 ms per each large packet.

Energy consumption

Energy consumption is one of the crucial issues for battery-powered wireless sensor motes. To maximize lifetime of, e.g., remotely placed sensors, all operations running on them should aim at minimizing the amount of consumed energy. For some sensor

applications such as, e.g., remote environmental monitoring or military networks, replacing the source of power (a battery) after initial deployment is complex if not impossible. Considering a security protocol for a WSN such as HIP, it is important to assess how efficient in terms of energy consumption its components are. Energy analysis has to evaluate the cost of both on-sensor processing and data transmission.

Table 7.4 presents energy consumption for different operations on an Imote2 sensor node running at 416 MHz. By measuring duration of particular operations and current draw during their execution we were able to calculate the amount of energy spent on these operations. The respective formula is

$$E = I \times U \times T,$$

where I is the current draw, U is the voltage (4 V in our case), and T is the execution time of an operation. The current consumption numbers in Table 7.4 are the median values of respective sets of measurements done with an external multimeter. The number of measurements in each set varies for different operations. For most operations we collected hundreds of current measurements (100, 500, 1000 and even 3000 values for some functions) but for certain operations the number of repetitions was limited to tens of values. However, limited number of current measurements with a certain degree should not have produced completely inaccurate or imprecise results. According to our analysis, most of the conducted current measurements are fluctuating insignificantly. For instance, for a *DSA create key* function we collected 3000 measurements and took their median as a reference value. Then taking samples of different size (5, 10, 50, 100, 500, 1000 etc.) from this "population", we concluded that their medians are close to each other (especially starting from a 50-value sample) and to our reference value. For a *RSA sign* function we collected 1000 measurements and defined their median value 153.6 A as a reference point. Then, 50, 100, 500-value samples taken randomly from these 1000 measurements had the median value varying from 153.5 to 153.6 A, which is again near the chosen reference point. These examples confirmed certain stability of our current measurements.

It should be noted that the presented values correspond to the energy consumed by all applications running on a device, i.e. the whole Imote2 system. The term *idle* is used in the table to account for an operation mode, when no applications but system processes are running. As results show, an average current draw in the idle state equals to 73 mA with the active radio module (module inserted into the kernel and radio interface is up) and 52 mA with the deactivated radio module. Thus, a real energy consumption overhead of an operation should be seen as the difference between the current draw value presented in Table 7.4 for the operation in question

Table 7.4: Energy consumption on a 416-MHz Imote2. Voltage 4 V.

OPERATION	**CURRENT** (median)	**DURATION** (median)	**ENERGY**
Idle, radio on	73 mA	N/A	N/A
Idle, radio off	52 mA	N/A	N/A
Booting Imote2 kernel	99 mA	23000.0 ms	9108.0 mJ
RSA-1024 create key	144 mA	2300 ms	1324.8 mJ
RSA-1024 sign	154 mA	20.0 ms	12.3 mJ
RSA-1024 verify	144 mA	1.1 ms	0.6 mJ
DSA-1024 create key	151 mA	12100 ms	7308.4 mJ
DSA-1024 sign	153 mA	11.2 ms	6.9 mJ
DSA-1024 verify	150 mA	12.0 ms	7.2 mJ
ECDSA-160 sign	141 mA	3.5 ms	2.0 mJ
ECDSA-160 verify	147 mA	9.4 ms	5.5 mJ
DH-1536 create	158 mA	349.5 ms	220.9 mJ
DH-1536 shared	158 mA	348.4 ms	220.2 mJ
ECDH-192 shared	147 mA	16.1 ms	9.5 mJ
SHA1 hash create	142 mA	0.03 ms	0.02 mJ
Polynomial $GF(2^{16}+1)$	150 mA	0.45 ms	0.27 mJ
Puzzle solving K=10	142 mA	28.5 ms	16.2 mJ

and the current draw in the idle state. This fact also downgrades the concern about the accuracy of our current measurements since we intend to show the difference between two values measured with the same method.

A reader should take into account that Table 7.4 does not contain current draw in a *sleep* and a *deep sleep* mode. Although Imote2 can operate in these low-power modes with current draw as low as 390 μA (see specifications in [2]), our Linux-based Imote2 sensor platform missed actual support of power management. In practice, each Imote2 node should be provided with due support of low-power modes and be able to deactivate its radio module and turn to a sleep state whenever it is not processing and transmitting data over a network.

With regards to energy cost of Imote2 data transmission over IEEE 802.15.4 radio, our results do not display a separate current draw for it. The reason is that according to our observations, sending or receiving packets over radio does not incur an additional cost by itself but rather draws the same level of current as with enabled radio module. In other words, the value of 73 mA should be interpreted as an average current draw level during Imote2's radio communications. An absolute amount of consumed energy will then depend on duration of data transmission (or, in fact, the size of protocol messages given a certain data rate), which, in turn, is defined by a protocol and a scenario in question.

7.4 Summary

In this chapter, we proposed to use HIP as a standardized protocol combining different mechanisms to address security issues of wireless sensor networks. To evaluate HIP computational complexity in WSNs, we ported the existing HIPL protocol implementation to the Linux-based Imote2 sensor platform. We then measured performance of the protocol components during a HIP base exchange between two sensors or a sensor and a server, as well as performed benchmarking of certain cryptographic operations involved in HIP on the Imote2 device.

Using a relatively precise external multimeter, we also measured current consumption by an Imote2 sensor node while running particular protocol components and cryptographic operations. We then calculated power consumption of these operations and, given their duration, estimated the amount of consumed energy. The energy consumption values presented in Table 7.4 serve as a good base for further analysis of applicability of different security components to wireless sensor networks. We postpone such analysis until Chapter 8 where we introduce several lightweight

alternatives to “standard” HIP components. The analysis therein contains three examples of WSN applications with different requirements, for which we identify necessary protocol features, their possible lightweight alternatives and compare BEX energy consumption in both approaches.

8 Lightweight alternatives to HIP components

In the previous chapters we evaluated performance of several IP security mechanisms (involved in HIP) on different types of resource-constrained devices including a Linux-based Nokia Internet Tablet, a Symbian S60 smartphone, a Wi-Fi access router running OpenWrt and an Imote2 wireless sensor platform powered by Linux. The analysis of the results shows that HIP in its existing form is feasible to use on these devices but only in certain cases. Depending on requirements of a particular application, individual security components might be either too expensive for a lightweight client or simply unnecessary in some scenarios and applications. This necessitates to look for alternative security mechanisms that can result in a more efficient protocol that will be suitable for distinct circumstances including the most constrained devices and scenarios.

In this chapter, we evaluate possible alternatives to existing HIP components that can make the protocol flexible and adaptable to a variety of applications. Our goal is to collect and analyse a combination of mechanisms that can be utilized as a security framework for multiple application scenarios with different types of devices and requirements. In the following sections we describe several existing lightweight security mechanisms that can serve as potential "substitutes" to original primitives and components of HIP. Among others, they include Elliptic Curve Cryptography (ECC), lightweight certificates, a polynomial key exchange and a recently proposed "HIP Diet Exchange (DEX)".

In Section 8.1 we briefly describe "Lightweight HIP" or "LHIP" [43, 44], an extension of HIP that aims to make the protocol suitable for low-power mobile devices by eliminating cost of PKC signatures. The section that follows presents performance of ECC-based cryptography primitives measured on Nokia N810 Internet Tablet and Intel Mote 2 sensor platform and serves as an indicator of ECC benefits over "traditional" PKC. In Sections 8.3 and 8.4 we describe two existing "lightweight security" approaches that make use of hash chains and Merkle trees for building efficient lightweight certificates and predistributed polynomial shares for implementing a fast key exchange. Section 8.5 presents in a nutshell a recently proposed "HIP Diet EXchange" or "HIP DEX". Finally, in Section 8.6 we provide our estimations of energy consumption of several protocol configurations consisting of the above presented components and show that they can significantly improve energy efficiency of HIP or a similar security protocol. The analysis is based on requirements of three examples of WSN applications and our measurements of power consumption on the

Imote2 sensor platform. The analysis demonstrates suitability of unmodified and modified HIP versions to scenarios with different requirements.

The ideas, empirical results and their analysis presented in this chapter were in portions published in the articles *Elliptic Curve Cryptography (ECC) for Host Identity Protocol (HIP)*[21] [122] and *On Application of Host Identity Protocol in Wireless Sensor Networks*[22] [67].

8.1 "Lightweight HIP"

Lightweight HIP or LHIP [43, 44] derived from early thoughts on computational complexity of public key cryptography in HIP when used on resource-constrained mobile clients. The motivation for this work was further reinforced later with availability of the first empirical HIP measurement results on Nokia 770 Internet Tablet [69]. In LHIP, 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, thereby making the protocol suitable for mobile appliances with constrained resources [43, 44]. However, while aiming to provide efficient secure mobility and multihoming for tiny devices, LHIP does not intend to achieve the same level of security as HIP does and accordingly has a number of limitations. First, assuming the absence of MitM attacks during BEX, LHIP does not use RSA/DSA signatures in $I2$ and $R2$ packets by default and, thus, it does not authenticate the host identities of communicating hosts (except of cases of namespace conflicts). Second, LHIP does not use DH nor does it establish a shared symmetric key between two end hosts, thus, it cannot provide payload encryption as in HIP. Finally, the use of interactive hash chains to authenticate LHIP update messages to communicating peers and to middleboxes increase communication overhead, especially in the presence of long RTTs [43, 44].

For the subject of this thesis, LHIP represents an essential reference work because it analyses computationally expensive components of HIP, discusses several options for

[21] © 2010 IEEE. Portions reprinted, with permission, from O. Ponomarev, A. Khurri, and A. Gurtov. 2010. Elliptic Curve Cryptography (ECC) for Host Identity Protocol (HIP). In: Proc. of the 9th International Conference on Networks (ICN 2010), pages 215–219. IEEE.

[22] © 2010 IEEE. Portions reprinted, with permission, from A. Khurri, D. Kuptsov, and A. Gurtov. 2010. On Application of Host Identity Protocol in Wireless Sensor Networks. In: Proc. of the 7th IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS'10), pages 358–365. IEEE.

their removal and finally proposes a lightweight authentication extension to reduce cost of PKC. While designing LHIP, the author shows that in order to keep a similar to HIP security level, it is impossible to remove PKC signatures or DH and it is not desirable to use shorter keys as they will reduce the strength of the protocol [43, 44]. This analysis illustrates clearly that to construct an equivalent in security but less expensive variant of the protocol other alternative mechanisms are required. In the following sections we analyse several alternative security mechanisms that notably decrease the price of "traditional" PKC algorithms used for digital signatures and key negotiation.

8.2 Elliptic Curve Cryptography

ECC is often treated as a "new generation" of public-key cryptography. It provides similar level of security as, e.g., RSA and DSA with a much shorter key length [16], thus fastening computations and requiring less memory space and bandwidth. Hence, the use of ECC is an attractive option for battery-powered resource-constrained devices. Having presented basic details of ECC in Chapter 2 and related work in Chapter 3, in this section we intend to show how ECC can improve performance of security protocols in practice on real devices.

We compare performance of "traditional" RSA [125], DSA [103] and DH [20] cryptography algorithms with their ECC counterparts such as ECDSA and ECDH [16] in the context of the Host Identity Protocol. So far, HIP specification [98] defines RSA and DSA as the main algorithms for digital signatures, and DH as the protocol for establishing a shared secret. However, the Host Identity can be constructed from an ECC key enabling the use of the ECDSA algorithm for signing HIP control messages, meanwhile replacing the expensive DH protocol with an efficient ECDH key exchange.

To measure performance of RSA, DSA and ECDSA signature operations, we used the OpenSSL *speed* utility version *1.0.0-beta3*. The utility was invoked multiple times on both devices (N810 and Imote2), each time doing a particular operation (e.g., *sign* with a 160-bit key) for approximately 10 seconds and then calculating the number of operations per second and the time needed for one operation. To measure duration of *DH_compute* and *ECDH_compute* operations, we used both OpenSSL and a benchmarking utility from the HIPL code, which did each of the operations for 90 seconds and calculated the average time out of that.

8.2.1 ECC on Nokia N810 Internet Tablet

Below we present our benchmarking results of different ECC primitives on a Nokia N810 Internet Tablet and provide a quantitative estimation of the ECC impact on the HIP BEX.

Signature operations and key agreement

Table 8.1 contains duration of RSA and ECDSA signature operations with different key lengths measured on a Nokia N810 Internet Tablet powered by a 400-MHz ARM processor and running the Linux kernel version 2.6.21. The ECC key lengths and performance numbers in Table 8.1 correspond to elliptic curve domain parameters over a prime finite field Fp recommended by the Standards for Efficient Cryptography Group (SECG) [16] (for *secp160r1*) and by NIST [103] (for *nistp192-nistp384*).

As was mentioned in Chapter 2, computational cost per bit (of the key length) of ECC signatures grows much slower than of similar RSA operations. Simple calculations on the numbers from Table 8.1 prove this statement – while changing the security level from 1024-bit to 2048-bit key (or, from 160-bit to 224-bit key), the computational cost per bit rises by 1.5 times (or 50%) for the *ECDSA sign* and by 3.4 times (or 239%) for the *RSA sign* operations. This demonstrates that benefits of using ECC will grow in the future when security requirements for communication systems will necessitate the use of long encryption/decryption keys.

Table 8.2 shows how much time a N810 consumes for computing a shared secret using different key lengths/groups with the "classic" DH and the ECC-based ECDH algorithms. ECDH clearly outperforms DH, both with small and big keys and therefore is an excellent option for resource-constrained devices saving their power and memory storage. While to compute a DH shared secret using a 2048-bit key a N810 requires 576.6 ms, a similar in security level ECDH operation using a 224-bit key takes on the Internet tablet only 16.3 ms.

Estimated BEX performance

The results presented in the previous paragraph allow us to estimate BEX duration on a Nokia N810 Internet Tablet with various key sizes. The current "default" configuration, which is used, for instance, in the HIPL code [1] is a combination of a 1024-bit RSA Host Identity for authentication and a 1536-bit group for the DH exchange. Taking this configuration as a reference point in our estimations, we then

Table 8.1: Signature operations at N810. Adapted from [122].

Algorithm and key size	**Sign**	**Verify**
RSA using 1024-bit key	24.0 *ms*	1.4 *ms*
RSA using 2048-bit key	160.8 *ms*	5.0 *ms*
RSA using 4096-bit key	1177.8 *ms*	19.3 *ms*
ECDSA using 160-bit key	2.6 *ms*	9.4 *ms*
ECDSA using 192-bit key	4.2 *ms*	18.7 *ms*
ECDSA using 224-bit key	5.5 *ms*	27.0 *ms*
ECDSA using 256-bit key	6.7 *ms*	33.3 *ms*
ECDSA using 384-bit key	15.5 *ms*	83.8 *ms*

replaced DH with ECDH of equivalent strength (see Table 2.1) that allowed us to keep an RSA Host Identity for compatibility with the current standard [98] but to spend less time for generation of a session key assuming that both parties support ECC. As a second step in our estimations, we replaced the RSA Host Identity with an ECDSA key, assuming that this change would be valid only if compatibility with legacy RSA/DSA systems is not needed.

The above configuration changes and transformation of cryptographic components resulted in three variants of BEX for each security level. We depict them, as well as the estimated BEX duration for a N810 acting as a client, in Table 8.3. While for a HIP server an important performance aspect is the number of incoming HIP connections it can handle, e.g., per second, for a resource-constrained client such as N810 the estimated BEX time provides a good indicator of the delay for interactive applications caused by applying this security mechanism. The numbers in Table 8.3 show that the duration of a HIP association establishment is reduced from 275 ms ("default" configuration) to 39 ms and further to 33 ms by subsequently applying ECDH and ECDSA.

The estimated BEX was calculated by adding duration of PKC signatures and DH key generation (taken from Tables 8.1 and 8.2) assuming that N810 acts as the

Table 8.2: Diffie-Hellman exchange at N810. Adapted from [122].

Algorithm and key length	**Duration**
DH compute using 768-bit group	33.9 ms
DH compute using 1536-bit group	248.4 ms
DH compute using 2048-bit group	576.6 ms
ECDH compute using 160-bit key	7.7 ms
ECDH compute using 192-bit key	11.8 ms
ECDH compute using 224-bit key	16.3 ms
ECDH compute using 256-bit key	21.3 ms
ECDH compute using 384-bit key	69.1 ms

initiator of the HIP connection, i.e. the respective formula was:

$$BEX = T_s + 2 \times T_v + T_{DH},$$

where T_s is the time needed to sign a message, T_v – the time needed to verify a message, and T_{DH} – the time to compute a DH session key. Hence, the presented estimation of BEX duration corresponds purely to the most expensive cryptographic components it involves. Accordingly, the numbers do not include any other protocol operations such us puzzle processing, generation and verification of HMAC signatures, building and handling HIP control packets (including those that close the HIP connection) and RTTs, which vary from network to network. However, even without the above operations the presented estimation suffices for achieving our goal – to compare performance of "traditional" PKC algorithms in BEX with their ECC counterparts. Assuming that other BEX components will remain in place, the values in Table 8.3 provide a good estimation of the difference in BEX performance caused by the use of ECC. The estimated BEX durations are rounded down to whole numbers.

The estimations indicate clearly that the DH key negotiation protocol is a major factor in duration of BEX, and by substituting it with a far more efficient ECDH algorithm the BEX delay is reduced tremendously (by 236 ms in this case). Replacing an RSA identity with an ECDSA identity produces a smaller effect decreasing

Table 8.3: Estimated BEX duration on N810. Adapted from [122].

Authentication	**Session key**	**BEX duration**
RSA-1024	DH-1536	275 ms
RSA-1024	ECDH-192	39 ms
ECDSA-160	ECDH-192	33 ms
RSA-2048	DH-2048	747 ms
RSA-2048	ECDH-224	187 ms
ECDSA-224	ECDH-224	76 ms

the BEX time further only by 6 ms. However, the positive impact of applying ECC to BEX rises with the growth of security level. The use of ECDH and ECDSA with a 224-bit key instead of a 2048-bit DH and RSA cuts the BEX duration down from 747 to 187 (by 560 ms) and further to 76 ms (by 111 ms) (see Table 8.3). In another representation, the use of ECDH instead of DH with this security level reduces BEX to 25% of the original BEX duration, while further substitution of RSA by ECDSA makes this modified BEX equal to 10% of the original BEX.

N810 – summary

The above measurements and estimations demonstrate how ECC can improve performance of HIP BEX on Nokia N810 Internet Tablet, a lightweight client with constrained resources (our original work [122] also presents the impact of ECC on a server processing incoming HIP connections). While using ECC for the Diffie-Hellman exchange, the compatibility with RSA/DSA-only systems persists, as the protocol allows the responder to offer the initiator both DH and ECDH options, and this speeds up BEX by several times. The use of an ECDSA identity instead of an RSA identity breaks compatibility with non-ECC systems and improves BEX performance notably only in the presence of high security requirements that necessitate using greater key lengths (over 2048 and 224 bit). Hence, while HIP can be modified to use ECDH as the key negotiation protocol even in the current HIP hosts, the use of ECDSA host identities requires to add their support to all communicating parties

Table 8.4: Duration of signature operations on Imote2.

		Average time (ms)					
Key length (bits)		**RSA**		**DSA**		**ECDSA**	
R(D)SA	ECC	*sign*	*verify*	*sign*	*verify*	sign	verify
512	**N/A**	4.1	0.3	4.2	3.8	*N/A*	*N/A*
1024	**160**	20.0	1.1	11.2	12.0	3.5	9.4
2048	**224**	127.1	3.9	38.2	43.9	6.7	26.4
3072	**256**	*N/A*	*N/A*	*N/A*	*N/A*	8.5	34.5
4096	**N/A**	911.1	14.9	*N/A*	*N/A*	*N/A*	*N/A*
7860	**384**	*N/A*	*N/A*	*N/A*	*N/A*	17.1	81.3

and, thus, is worth implementing it in pure ECC-based systems with high security threats.

8.2.2 ECC on Imote2 sensor platform

In the paragraphs below we present measurement results of ECC signature and DH operations on the Imote2 wireless sensor platform. The results are shown in comparison with other PKC algorithms. Although performance of these cryptographic primitives on Imote2 is similar to the presented above results measured on Nokia N810 Internet Tablet (probably because they are equipped with a similar class of hardware), we find it important to report both sets of measurements as they were performed on yet different types of devices and can be utilized as a reference in future research on mobile Internet and WSNs.

ECDSA signatures and verifications

Table 8.4 contains average times needed to generate and verify a signature on the Imote2 sensor platform using RSA, DSA and ECDSA keys of different length. The comparison of "traditional" asymmetric algorithms such as RSA and DSA with El-

liptic Curve Cryptography (ECC) is especially interesting since ECC achieves the same level of security with notably shorter keys [16], thus providing better performance in terms of faster computations, saved energy and memory space. Please note that for particular RSA/DSA key lengths (such as 512 and 4096 bits in) there is no equivalent ECC key length, which is indicated in Table 8.4 as "N/A". The same sign indicates the absence of measurements for certain algorithms and key lengths. As in the N810 case, the ECC key lengths in Table 8.4 correspond to elliptic curve domain parameters over a prime finite field Fp recommended by SECG [16] and NIST [103].

As stated before, benefits of ECC are increasing with rising the level of security, i.e the key length. Compare, for instance, performance of sign operations done with a 4096-bit RSA key (911.1 ms) and an even more secure 384-bit ECDSA key (17.1 ms). Although we were not able to measure operations with all possible key lengths that in the RSA/DSA and ECDSA algorithms directly correspond to each other in security level, the results in Table 8.4 indicate well the speed of computational delay's growth with increasing the key length or changing from the previous security level to the next. This speed is notably smaller with ECDSA – for instance, the duration of the *ECDSA sign* operation grows from 3.5 ms (using a 160-bit key) to 6.7 ms (using a 224-bit key) only by 1.9 times, whereas an equivalent change of the security level makes the *RSA sign* to increase by 6.4 times, from 20 ms with a 1024-bit key to 127.1 ms with a 2048-bit key.

In addition, it is important to note the difference in performance of *sign* and *verify* operations in different algorithms. While verification of RSA signatures takes significantly less time than their generation, DSA sign and verify operations are of the same order of magnitude. ECDSA, in turn, has a different behaviour with verification of signatures consuming more time than their generation. The computational imbalance of sign and verify operations (and thus decryption and encryption operations), especially in the RSA algorithm, should be taken into account while designing secure communication systems involving devices with different computational power and hardware resources.

ECDH shared key computation

Table 8.5 illustrates duration of shared key computation using the "conventional" DH and the Elliptic Curve DH (ECDH) algorithms. Our measurement results indicate that with a 1536-bit DH group it takes for an Imote2 sensor mote on the average 349.5 ms to create a DH public value and 348.4 ms to generate a shared secret key,

Table 8.5: Duration of shared key computation on Imote2.

Protocol	**Average time (ms)**
DH compute 768	89.7
DH compute 1536	348.4
ECDH compute 160	7.7
ECDH compute 192	16.1
ECDH compute 224	21.7
ECDH compute 256	28.9
ECDH compute 384	66.6

resulting in an overall 697.9 ms-delay for a complete DH exchange. In contrast, a corresponding (in security level) ECDH exchange with a 192-bit ECDH key is accomplished on Imote2 as fast as in 32.2 ms on the average. With increasing the key length, the benefits of ECC are growing.

Imote2 – summary

The measurement results presented above show potential performance gains of ECC in the HIP BEX that can be achieved on the Imote2 sensor platform. With presently "typical" security requirements, the use of the ECDH algorithm reduces the computational cost of key negotiation to 5% of the cost of "classical" DH (compare numbers for *DH-1536* and *ECDH-192*). Using the ECDSA algorithm instead of RSA or DSA provides more performance benefits to Imote2 with raising the level of security. For instance, the cost of *ECDSA sign* drops from 18% to 5% of the *RSA sign* cost while increasing the key length from 1024 to 2048 bits for RSA or from 160 to 224 bits for ECDSA.

8.3 Lightweight certificates

So far, HIP has been specified for using only asymmetric cryptography for operations on digital signatures and negotiation of a shared symmetric key [98], which on the one hand provides scalability and perfect security, but on the other hand requires more computational, time and energy resources. As less scalable but more efficient alternatives for particular types of networks (involving extremely constrained devices and tight operational requirements), several research projects proposed to use one-way secure hash functions, Merkle trees and evaluation of predistributed polynomial shares to implement efficient authentication of nodes and establishment of a pairwise shared secret between any pair of nodes [21, 35, 34, 86, 85]. In this and the following section we will present basic details of these approaches applied to HIP and intended for resource-constrained wireless sensor devices. The approaches presented below were analysed in our previous work [67].

Although public key certificates presented in Section 7.2.1 allow to fulfil the authentication and authorization security requirements of mobile and wireless sensor networks in a scalable and flexible way, in certain application scenarios, such as a static unattended WSN or an on-body low-power personal WSN communicating through a gateway, introducing a complex PKI is challenging, resource-demanding and does not bring extra benefits. Omitting a PKI and introducing an alternative mechanism to verify digital certificates would allow to avoid cost of PKC, while still supporting authorization and authentication of host identities.

Lightweight certificates based on one-way hash functions and Merkle trees have received close attention in research [21, 35]. In this section, we use a notion of a *lightweight certificate* as it is defined in [34]. One of the representations is based on a Merkle tree [92, 91]. Such certification structure is a binary tree, where each of n leaves L_i is calculated as the hash of a value $HIT_i \| Role_i \| Time_{validity}$, and each internal node m_{ij} is calculated as the hash of the concatenation of its two sibling nodes $m_{ij} = H(m_i || m_j)$. The root of the tree together with $log_2(n)$ elements can be used to verify any leave in the tree.

The above information from the binary tree combined with the corresponding path is sufficient to allow any node (which has the root element of the tree) to efficiently perform (i) authentication of an identity (a public key), (ii) authorization of any other node that holds the corresponding identity, and (iii) verification of any other additional information bound to the identity. Therefore, the advantage of lightweight certification is that it provides efficient authentication and authorization of an identity and allows to build access control mechanisms without heavy

computations typical for PKC approach.

In practice, such lightweight certificates (generated similarly to PKC certificates by a trusted CA) can be easily integrated in HIP, as the approach described in [48] has no limitation on the type of certificates carried in $R1$, $I2$, or $R2$ signalling packets.

8.4 Polynomial key exchange

The idea of using bivariate polynomials for deriving a shared secret key has been originally proposed in the works by Blom [10] and Blundo *et al.* [11] and later applied to other systems including wireless sensor networks [35, 34, 86, 85]. Here, we describe the proposed technique briefly, based on related work. Our suggestion (previously presented in [67]) is that a polynomial exchange can serve as an efficient alternative to the computationally expensive Diffie-Hellman protocol for key negotiation in HIP in wireless sensor networks.

The approach in question makes use of a t-degree bivariate polynomial $f(x, y) = \sum_{i,j=0}^{t} a_{ij} x^i y^j$ over a finite field $GF(q)$, such that q is a large enough prime to accommodate a cryptographic key. The bivariate polynomial has the following property $f(x, y) = f(y, x)$ and is generated by a trusted server. Besides generating the polynomial itself, the server calculates a polynomial share for each node i by evaluating the original bivariate polynomial at $x = i$, where i is the identity of the node. The corresponding share $f(i, y)$ is preloaded onto the node i prior to its deployment [85]. The technique allows any two nodes i and j to derive a pairwise secret key by evaluating their polynomial shares further using each other's identities as points $K_{ij} = f(i, y = j) = f(j, y = i) = f(i, j)$ [35, 86, 85].

Storing all polynomial shares on each sensor node provides a good secrecy and ensures successful key negotiation. However, if a network comprises several thousands sensor nodes, for example, a grid-based assignment [85] can be used to minimize memory footprint, as well as knowledge of deployment/location [21, 86] to minimize communication overhead. This will guarantee (i) that any pair of sensors can agree on a common secret, and (ii) resilience to a node compromise when certain conditions are met. For further details please refer to [86, 85]. A lightweight version of the HIP base exchange using preshared polynomial shares as a way to derive a common secret is shown in Figure 8.1.

The parameters in square brackets in Figure 8.1 are optional meaning that (i) PKC signatures can be present if compatibility is required, (ii) lightweight certificates can

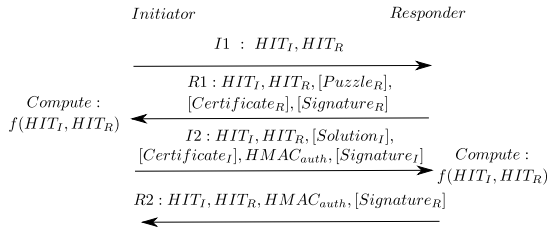


Figure 8.1: Lightweight HIP base exchange.

be carried if the network needs to support node authorization. Puzzle parameters are used when there is a DoS threat and for negotiating random values I and J used in KDF.

During the BEX in Figure 8.1 both initiator and responder use a modified key derivation function (KDF) [98] shown in Equation (8.1). Here, the *postfix* is defined as $sort(HIT_I \| HIT_R) \| I \| J \| 0x01$, where I and J are nonce values, and *sort* is a function that sorts HITs as specified in [98]. Thus, to obtain the k^{th} key, nodes evaluate (8.1) k times starting from the initial key K_0.

$$K_k = HASH(HASH(K_{k-1}) \| postfix) \tag{8.1}$$

According to the original work [11], the schemes relying on evaluating t-degree polynomials guarantee resistance to no more than t compromised nodes that cannot obtain information about the pairwise key established between any two non-compromised nodes. Different optimization techniques for these schemes have been proposed to minimize memory usage and communication overhead for wireless sensor nodes [86, 85].

8.5 HIP "Diet EXchange"

Moskowitz has recently proposed a lightweight version of the HIP BEX called "HIP Diet EXchange" (or "HIP DEX") [96]. DEX is intended for devices that have limited processor power and storage. DEX aims to decrease computational delay of HIP association establishment by replacing certain HIP components with their efficient alternatives, by using as few cryptographic primitives as possible and by providing other optimizations to the protocol taking into account constraints of such devices

and networks they usually operate in. In general, DEX is similar to BEX in that it is a four-packet exchange, does use the same mechanisms for updating and closing an association etc. and can be used with IPsec ESP or other protocol to protect payload data. However, HIP DEX has a number of important differences from HIP BEX, which are summarized below [96]:

- The DH exchange is replaced with the ECDH exchange, the RSA/DSA algorithms are replaced with the ECDSA algorithm;
- DEX uses AES-CBC (Cipher Block Chaining) encryption algorithm mode to provide CMAC that substitutes HMAC used in BEX;
- Accordingly with the above, the puzzle mechanism uses CMAC instead of HMAC;
- A HIT (Host Identity Tag) is not generated by applying a hash function to the public key of the host, instead a HIT is a 96-bit truncated Elliptic Curve Host Identity that, in turn, is a fixed private (ECDH) key of the host generated from a random number;
- DEX does not provide anonymity since there is no option for encryption of Host Identities of HIP initiator and responder;
- Because DEX is expected to be used on lightweight devices operating in wireless networks with high rate of packet loss, DEX introduces a new packet retransmission mechanism where the HIP DEX initiator is supposed to be sending $I1$ and $I2$ messages at a continuous rate until it receives the corresponding $R1$ and $R2$ packets;
- DEX does not provide perfect forward secrecy and, thus, is exposed to the risk of getting all HIP associations compromised once the host identity of the HIP initiator used in these associations has been compromised.

Further details of HIP DEX can be found in the respective Internet draft [96].

8.6 Analysis and applications

In this section we present an analysis of HIP "standard" and alternative components applied to three wireless sensor network (WSN) application types (see Table 8.6):

(1) a battlefield sensor network representing a "critical" WSN class with a maximum set of requirements; (2) a body sensor network representing a "hybrid" WSN class and consisting of two sub-networks, a PAN (Patient Area Network) and a BEN (Back-End Network), as they defined in [34]; and (3) a sensor network used for non-critical environmental monitoring (e.g., to sense humidity, temperature etc.) that in our example represents a "lightweight" class of WSNs.

Table 8.6 contains a summary of our analysis by listing in the middle a set of requirements for WSNs, on the left side – a set of standard protocol components and their alternatives, and on the right side – our three example WSN applications. The main alternatives used in building different protocol configurations are ECC-based algorithms for signature operations and shared secret negotiation, lightweight (LW) certificates instead of PKC certificates and a preshared polynomial key exchange as a further possible replacement to a DH or a ECDH exchange.

To show how WSN requirements correspond to protocol components and application types, in Table 8.6 we use "x" to indicate a mandatory option; "[x]" to indicate that either of two components with this sign can be used to satisfy a correspondent requirement; "(x)" to indicate that a "polynomial exchange" alone is sufficient to meet "authentication" requirement; and "(x)*" to indicate that "confidentiality" is not mandatory for "Monitoring" (in our scenario) but is already fulfilled by the use of a "polynomial exchange".

As a part of our analysis, for each application we estimated energy cost per BEX for both HIP initiator and responder, each being an Imote2 sensor mote. We computed the energy cost of BEX by adding energy cost of the main BEX components:

$$BEX_{Battlefield} = PKCcert + PKCsig + Puzzle + DH + HMAC,$$

where:

- $PKCcert$ equals to one PKC signature verification;
- $PKCsig$ include one signature and two verification operations for HIP initiator and vice versa for HIP responder;
- $Puzzle$ implies solving a puzzle for HIP initiator and checking the solution (equal to one hash operation) for HIP responder;
- DH is composed of one DH_{create} and one $DH_{compute}$ operation for both HIP hosts;

Table 8.6: Protocol components needed for different WSN applications (based on identified requirements) and their energy cost per BEX for HIP initiator and responder. Numbers in the round brackets are the cost of BEX using RSA and DH ("standard" HIP). Numbers before the brackets are the cost of BEX using ECDSA and ECDH. Numbers for "Bodynets PAN" and "Monitoring" (1.5 mJ) are the cost of BEX based on LW certificates and a polynomial exchange. Adapted from [67].

Protocol components							**Requirements**	**WSN application types**			
Certificates		*Signatures*		*Puzzle*	*Key exchange*			*Battlefield*	*Bodynets*		*Monitoring*
PKC	LW	RSA	ECDSA		DH	Poly		(critical)	PAN	BEN	(lightweight)
[x]	[x]	[x]	[x]			(x)	Authentication	x	x	x	x
[x]	[x]						Authorization	x	x	x	x
					[x]	[x]	Confidentiality	x	x	x	(x)*
		[x]	[x]				Integrity	x	x	x	x
				x			DoS resistance	x		x	
x		[x]	[x]		x		WSN scalability	x		x	
	x					x	WSN lifetime		x		x
							Energy/I, mJ →	53.8 (471.4)	1.5	53.8 (471.4)	1.5
							Energy/R, mJ →	34.1 (467.0)	1.5	34.1 (467.0)	1.5

- $HMAC$ corresponds to $HMAC_{sig}$ and $HMAC_{ver}$, each being equal to two hash functions (SHA1 in our case).

The corresponding formula for HIP BEX with lightweight components is:

$$BEX_{PAN} = LWcert + PolyEx + HMAC,$$

where:

- $LWcert$ corresponds to checking one LW certificate, which is equal to one hash operation;
- $PolyEx$ is a polynomial exchange that provides authentication, confidentiality and integrity (also replacing DH) and is equal to five $GF(2^{16} + 1)$ polynomial operations;
- $HMAC$ is as above.

In our scenarios, the $Bodynets_{BEN}$ application has similar requirements to the $Battlefield$ application, whereas $Monitoring$ in its requirements is similar to $Bodynets_{PAN}$, which is reflected in calculation of their BEX energy cost.

The results presented in Table 8.6 assume the use of ECC-based PKC signatures and certificates and DH (ECDSA and ECDH) for the most critical "battlefield" application and, in turn, LW certificates and a polynomial exchange for the least critical "monitoring" application. For estimations we relied on BEX versions depicted in Figures 2.1 and 8.1 and used energy consumption values from Table 7.4. We did not include into BEX estimation two round trip times (RTT) since their value varies for different packet sizes and, in our experience, also depends on the quality of the radio driver and the signal level. In our experiments, RTT varied from 13 to 127 ms on the average for packets ranging from 64 to 1028 bytes. Although the puzzle in our scenarios is unlikely to be used, we kept it in calculations.

Overall, our measurements (Table 7.4) and estimations (Table 8.6) demonstrate a significant difference between the computation and energy cost of PKC and LW approaches. The energy cost of the most "lightweight" BEX variant in our scenarios (using LW certificates and a polynomial exchange) is equal to only 2.8% (for the HIP initiator) and 4.4% (for the HIP responder) of the cost of the BEX using ECDSA and ECDH, and to just 0.3% of the "standard" HIP BEX using conventional RSA and DH algorithms. From another angle, the estimated ECC-based BEX consumes 11.4%

and 7.3% of the energy cost of the "classic BEX" for the HIP initiator and responder respectively. The presented combination of security mechanisms can serve as a solid framework useful in design, development and evaluation of flexible communications security protocols applicable to a variety of WSN scenarios. Given a real WSN deployment case with knowledge of particular security requirements and the use of certain protocol components, the presented results would allow easy assessment of the protocol performance and estimation of the network lifetime.

9 Summary and discussion

In this chapter we summarize and interpret the results of our empirical research. We present our view on feasibility of running existing IP security mechanisms on resource-constrained hardware and make recommendations on the use of unmodified HIP on such devices. In the second part of the chapter, we discuss the potential of several lightweight security techniques presented in the previous chapter and, based on our measurements and estimations, conclude on the trade-offs they can bring. In the end of the chapter, we present several prospective directions for future research.

9.1 HIP applicability to lightweight devices

While assessing the feasibility of running HIP on resource-constrained devices we discuss two approaches: (a) using the unmodified ("original", "standard" or "classic") protocol and determining the types of devices, applications and scenarios it can suit and (b) using lightweight alternatives (to certain computationally intensive security components) that would improve protocol performance on extremely constrained devices and/or in scenarios/applications with tight operational requirements.

9.1.1 Unmodified Host Identity Protocol

In general, the obtained results (presented in Chapter 5) 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 Nokia 770 and Nokia E51. However, this is a general statement and in practice the applicability of unmodified HIP as a secure mobility protocol to 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 a single HIP base exchange with the proxy server is sufficient for the whole browsing session (unless the mobile client changes its network attachment point), most users will probably tolerate this delay. We

make such conclusion based on the Nielsen's usability book [107], where the author elaborates on the "0.1/1/10" rule for the interactive applications, studied earlier by Miller [93] and Card *et al.* [15]. 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 [107]. 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 for a BEX between the phone and a server). Because the user of a mobile phone most probably uses HIP with a GUI application, which turns the phone into the *active* state, establishment of a HIP association will always produce a three-second delay for the users of smartphones such as 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 4 ms. Hence, the applications requiring the handoffs faster than this delay 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 2 ms. It is worth noting that these mobility measurements were performed in 2007 with a code snapshot available at the time, hence today's numbers

might slightly change with the present implementations. In general, however, 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 delay for an application introduced by HIP is to adjust different protocol 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 for signature operations. 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, with new requirements to future communications systems, the trend is to increase the level of security (i.e., the length of RSA and DH keys in this case) 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 the puzzle by the *Initiator* whereas raising K to 15 and above increases the processing time exponentially on the Nokia 770 Internet Tablet. As a potential protocol enhancement based on adjusting parameters, 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 lower puzzle difficulty.

Taking into consideration the processor utilization perspective, our results with a Symbian-based Nokia E51 indicate a close to 100% processor load during 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 a listening/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 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/median value). With the initial RTT=2.08 ms, the ESP encryption increased this value by 0.67 ms. It would be unfair to generalize these results because the RTT varies notably 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 unidentified reason. An average value of 4.86 Mbps appeared to be an upper bound for Nokia 770 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 in hardware, WPA expectedly produced a tiny impact as compared with 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 have potentially affected 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 the TCP protocol itself and by the network channel [132]. We leave such an 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 resulted in 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 work of a 1050-mAh battery. In reality, the constant exchange of the HIP control packets would be an 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.

Performance of HIP operations on Intel Mote 2, a wireless sensor platform, (pre-

sented in Chapter 7) is in the same range with the above results obtained on mobile devices. For instance, a single HIP base exchange between two Imote2 sensors communicating over IEEE 802.15.4 radio took in our tests 1.7 seconds with RTT varying (due to packet fragmentation and retransmissions) from 13 to 127 ms on the average for packets ranging from 64 to 1028 bytes. Processing of a cryptographic puzzle on Imote2 showed similar with mobile devices dynamics, increasing exponentially with the growth of the puzzle difficulty. Unlike experiments with Symbian-based phones, on the Imote2 sensor platform and on the Nokia N810 Internet Tablet we were able to perform isolated measurements of a set of cryptographic primitives and protocol components, such as operations on digital signatures, negotiation of a shared secret key, puzzle processing etc. These measurements allowed to determine the computational overhead of particular protocol components more precisely, as well as to compare different algorithms with each other. We also measured the current consumed by each primitive/function/mechanism separately, which allowed us to compare energy consumption of different protocol configurations including the variants using lightweight security mechanisms summarized in the next section.

9.1.2 Lightweight components as basis for adaptable security

With proliferation of new mobile services and wireless sensor network applications, requirements towards security level in these systems are constantly increasing. The use of traditional asymmetric cryptography as a part of underlying security protocols in the long term means an inevitable need to increase the length of a cryptographic key, which would result in longer computational delays and faster depletion of power – absolutely non-desirable consequences for resource-constrained devices. In Chapter 8 we presented details and performance analysis of several existing security mechanisms that are computationally efficient and thus appropriate for applications that either do not tolerate the complexity or do not require the strength of conventional PKC. Combined together, both traditional PKC approaches and their potential lightweight alternatives provide a framework that can be used in designing an adaptable security architecture. However, the present availability and the standardization and deployment status of lightweight mechanisms, are not fully transparent and need further investigation.

The potential of ECC and its status

Elliptic Curve Cryptography has a high potential for future data communications systems consisting of resource-constrained mobile and wireless sensor devices. ECC provides similar to traditional PKC algorithms security level, while operating on much shorter cryptographic keys [16, 105] (see also Table 2.1). This makes ECC computationally efficient, consumes less space for storing keys and signatures and speeds up transmission of control packets in bandwidth-limited networks. As a consequence, ECC reduces protocol operational time and saves energy crucial for a battery-powered device. National Security Agency (NSA) provides its estimations of the ratio of the DH cost to the EC cost based on the theoretical knowledge of cost of math operations, pointing out that performance of actual implementations may differ due to device architecture [105]. Related work presented in Chapter 3.1 gives other examples of ECC performance in different environments. Our empirical results in Sections 8.2.1 and 8.2.2 not only support theoretical knowledge about ECC gains and results from related work but also indicate performance level of ECC on concrete devices, a Nokia N810 Internet Tablet and an Imote2 wireless sensor platform. In particular, the measurements show that, compared with conventional asymmetric cryptography, ECC significantly reduces the cost of generating a digital signature and computing a shared secret, especially with high security levels. For instance, the ECDSA and ECDH operations (with a $224 - bit$ key) applied in BEX on N810 would consume only 10% of the computational cost of similar in security RSA and DH operations (with a 2048-bit key). On Imote2, an ECDH compute operation takes only 5% of the time required for a similar DH compute operation, while the performance improvement of an ECDSA sign function over an RSA sign operation notably grows along with increasing security level.

The reverse side of the "ECC medal", as was already mentioned in Chapter 2, is the uncertainty about the status of intellectual property rights (IPR) of ECC-based algorithms, methods and aspects. This uncertainty is two-fold with one dimension – the lack of standardization base – being closely related to, dependent on and conflicting with another dimension – the patents covering numerous ECC technologies [90]. Different companies and individuals around the world hold ECC-related patents. A variety of Internet resources discuss the IPR status of ECC and provide information about ECC patents. For instance, Google offers an opportunity to search and download text of all patents issued in the United States[23], while the Canadian company Certicom, the major holder of patents related to ECC and PKC

[23]Google Patents http://www.google.com/patents

in general [105], published a list of its patents and patent applications as of February 10, 2005 in a letter to the SECG consortium members[24]. However, in practice the process of standardization of ECC technologies is challenging. This complexity is reflected in the SECG patent policy that encourages members of the SECG consortium and other concerned parties to find out technologies that are patented and to work towards granting licenses to these technologies, which should facilitate their standardization and implementation[25]. So far, SECG has released two standards, SEC 1 [16] and SEC 2 [17].

According to NSA [105], despite complex IPR issues surrounding ECC, several countries around the world such as the US, the UK, Canada and particular NATO members have adopted different extents of ECC to protect sensitive government information in future communications systems. For instance, the US Department of Defence (DoD) is targeting to replace around 1.3 million existing hardware devices within a time frame of ten years. Furthermore, NSA has purchased from Certicom a license covering 26 ECC-related patents that among other documents are listed in a sample license between NSA and a potential company-sublicensee[26]. The purchased license has a number of restrictions including the use only for national security purposes and, as specified in the NIST FIPS 140-2 standard, limited to only prime field curves with the prime greater than 2255, which corresponds to the NIST curves with primes of 256, 384 and 521 bits. Vendors implementing particular commercial products within this field of use can obtain the license either from NSA or directly from Certicom [105].

ECC-based technologies covered by the above NSA license formed the core of Suite B – a set of cryptographic protocols and algorithms that have been identified to be necessary and sufficient to secure information up to the "SECRET" level. The suite contains a number of protocols specified by the IETF (as RFCs and Internet drafts) and approved by NIST (as FIPS standards). The ECDSA [103] and ECDH [102] algorithms are a part of Suite B (that also includes the AES and SHA algorithms) [105]. To facilitate the use of these algorithms, NSA has recently published two guides [106, 104] for implementers of the ECDSA and ECDH standards. McGrew *et al.*provide a background on mathematical operations and elliptic curve groups, as well as describe the ECDH and the Elliptic Curve ElGamal Signature algorithms mainly based on the original normative references to assist parties who wish to implement these algorithms without further methods [90].

[24]http://www.secg.org/download/aid-398/certicom_patent_letter_SECG.pdf
[25]SECG Patent Policy http://www.secg.org/index.php?action=secg,about_patents
[26]http://www.nsa.gov/business/_files/sample_license.pdf

As a summary, interest in ECC as a prospective substitution of classic PKC algorithms is growing. However, implementation and deployment of ECC-related technologies are still slow. In case a technology appears to be under a patent, parties considering to implement and include it in in their products usually either proceed with obtaining a license covering this technology (as NSA did) or buy an existing toolkit that allows to utilize ECC in a variety of applications (for instance, Certicom's Security Builder® NSE™ that covers 26 patents licensed by NSA). On the one hand, identification of patented aspects in practice is not a straightforward process and, even if those aspects are identified, not many vendors are ready to make large investments into licensing. These issues may prevent wide and fast adoption of ECC in the near future. On the other hand, companies naturally look for alternative non-patented implementations of the technology of their interest, which according to RSA Laboratories[27] might be worthwhile, unless the concept itself is patented. In arguable situations, solving conflicts might require a court assistance, as it was, e.g., in the case with Sony and Certicom[28]. Nevertheless, a number of companies utilize ECC in their commercial products[29], and there are grounds for this tendency to continue in the future.

Lightweight certificates and polynomial key exchange

The analysis of concrete WSN applications presented in Section 8.6 shows that substitution of expensive traditional PKC signatures and DH exchange in HIP with lightweight certificates based on one-way hash functions and polynomial-based key negotiation can drastically reduce the computational and energy cost of establishing a HIP secure association on an Imote2 wireless sensor mote. According to our measurements of current draw and duration of different cryptographic primitives and protocol components, the most "lightweight" version of BEX using one-way hash function-based certificates and a polynomial exchange would consume only 1.5 mJ per handshake for both the HIP initiator and responder, which is equal to 2.8% (for the initiator) and 4.4% (for the responder) of the cost of the ECC-based BEX (53.8 mJ and 34.1 mJ), and to only 0.03% of the "classic" HIP BEX using conventional RSA and DH algorithms (471.4 mJ and 467 mJ). The use of

[27] http://www.rsa.com/rsalabs/node.asp?id=2325

[28] http://www.certicom.com/index.php/2007-press-releases/32-2007-press-releases/20-certicom-f iles-suit-against-sony-for-patent-infringement

[29] Cryptomathic http://www.cryptomathic.com/Default.aspx?ID=316
Microsoft http://www.microsoft.com/windows/windowsmedia/forpros/drm/default.mspx
Nintendo http://www.nintendo.com/wii

lightweight certificates and polynomial-based exchange in HIP or a similar protocol as an alternative to PKC signatures and DH protocol for authentication and pairwise key establishment is, thus, especially beneficial for low-power battery-driven devices.

HIP "Diet EXchange"

A recently specified "HIP Diet EXchange" [96] is an important step forward towards making HIP suitable for embedded resource-constrained environments by using ECC instead of traditional PKC, CMAC instead of HMAC and further adjustments of protocol parameters. The process of standardization of this lightweight variant of HIP is essential for research community to develop it further and is a basis for actual protocol implementations. The measurements and estimations of ECC-based algorithms on certain devices presented in this thesis can serve as a reference in development of HIP DEX.

9.2 Possible future research

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" [54], 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 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 issues of lightweight communications systems. In particular, we empirically evaluated the applicability of the Host Identity Protocol and the IPsec protocol to four classes of resource-constrained devices including a Linux-based PDA, a Symbian smartphone, two models of the OpenWrt-based Wi-Fi access routers and the Imote2 wireless sensor platform. 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. For future research, we consider important to investigate this emerging concept further by focusing on its two domains: identification and access control for embedded objects and

secure communications between them. An existing work by Urien proposes an architecture for linking HIP-based active tags (RFIDs with tamper resistant resources and a securely stored identity) with HIP-enabled portals through IP networks [139].

An interesting recent initiative related to the Internet of Things is the industry alliance IPSO (IP for Smart Objects) [23]. 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 information, promotion and marketing entity [23].

The term "Internet of Things" is often used concurrently with the term "wireless embedded Internet". 6LoWPAN (IPv6 over Low Power Wireless Personal Area Networks) is a technology and the name of the IETF working group that aims to enable IPv6 communications over IEEE 802.15.4 radio networks on extremely resource-constrained, low-power, low-cost and small form factor devices. So far, the working group has produced two standards, RFC 4919 [79] and RFC 4944 [95], and a number of Internet drafts. A recent book [131] by Zach Shelby and Carsten Bormann is the first book on 6LoWPAN that introduces the details of the 6LoWPAN technology, explains its implementation aspects and illustrates application examples. A part of the book is devoted to security of 6LoWPAN. Both 6LowPAN and IPSO initiatives facilitate the use of TCP/IP technologies in wireless embedded networks.

Looking at the existing research trends in the field of communications security in wireless sensor networks (or networks composed of other lightweight objects), one essential issue remaining valid for future investigations is energy consumption. Maximized lifetime is often one of the crucial requirements for the networks operating on a remote side without a chance of replacing power supply on the motes once they have been deployed. While in this work we focused on measuring the effect of on-device computations on the battery lifetime of different handhelds and sensors, another factor strongly influencing power consumption of such devices is data transmission over a wireless interface. Ideally, a wireless device should stay most of the time in a sleep mode and wake up only for short periods to exchange data with its peers, as required by the secure communication protocol in question. Different approaches have been proposed to eliminate idle state of lightweight communication devices in wireless ad-hoc networks and, thus, to achieve rational utilization of their energy resources. A substantial contribution in this area valuable to future research on energy efficiency belongs to Feeney *et al.*. In their studies [26, 27, 28], the au-

thors looked into multiple power save protocols in wireless networks and evaluated the impact of different node wake-up algorithms on network capacity and energy efficiency.

As a potential continuation of our present research and a target for future work, we consider the wireless embedded Internet or the Internet of Things as an environment where different embedded devices (from a light bulb to a temperature meter) co-exist, 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 participants. To achieve this goal, we see a need for a combination of different security mechanisms that, being flexible and adaptable, would be able to meet security requirements of various application types in various scenarios on various classes of communication devices. Flexibility and adaptability should then allow to find the most efficient (in terms of computational complexity, operational and energy efficiency) combination of security mechanisms in each particular case, opting, for instance, between PKC signatures and hash chains or between an ECDH and a polynomial exchange. The mechanisms presented in this thesis and their evaluation can then serve as a framework for designing new protocols and their analysis in specific cases.

One particular piece of potential future work related to the subject of this thesis is to standardize (so that it becomes an RFC), implement, deploy and evaluate "HIP DEX" on various types of extremely constrained communication devices being prospective members of future wireless embedded networks and the Internet of Things.

10 Conclusions

In this thesis, we addressed the aspect of secure communications in the mobile and wireless embedded Internet consisting of various mobile devices and wireless sensor platforms. Our main objective was to evaluate feasibility of running existing network-layer security mechanisms (in particular, HIP and IPsec) on lightweight hardware and to assess the impact of limited computational, energy and network resources on performance of the security protocols.

We performed a literature study to obtain the insights into the research target and evaluated the contribution of related work. In the empirical part of the work, we conducted a number of experiments involving mobile handhelds, Wi-Fi access routers and wireless sensor platforms with low processor, storage and battery resources running the Host Identity Protocol. We measured performance of cryptographic primitives, HIP association establishment and mobility update, and the impact of IPsec encryption. The analysis of the results allowed us to make recommendations on using unmodified public key cryptography mechanisms implemented in software on low-power resource-constrained devices. In addition, we reported our development and porting experience, which can be useful for migration of OSS projects to an embedded platform.

Although unmodified IP security mechanisms proved to be applicable to resource-constrained devices in certain scenarios, in most cases optimizations to existing approaches and protocol modifications are highly desirable. For applications and scenarios that either do not tolerate or do not require strong cryptography, we evaluated a number of lightweight mechanisms that can serve as substitutes to the original "expensive" HIP components, making the protocol more efficient while maintaining its security on a level necessary for a particular application. To illustrate benefits of lightweight approaches in comparison with the traditional ones, we analysed three examples of wireless sensor network applications and estimated their energy consumption for establishing a secure association. Presented "standard" and "lightweight" components can potentially form a flexible and adaptable security framework to be utilized for multiple application scenarios with different types of hardware platforms, security, operational and QoS requirements.

Despite HIP is an end-to-end security protocol, its applicability to lightweight communication environments should be assessed not only by performance of individual end hosts (e.g., mobile terminals) but also from the view point of middleboxes and other infrastructural components. Our experiments with two different Wi-Fi access

routers used for piloting a HIP-based authentication architecture in wireless LANs showed that the low cost router model limited in processor power and memory is not capable of running all necessary security components, namely acting as a HIP-based firewall and a proxy for multiple clients. Hence, evaluation of HIP and other security protocols has to take into account the whole ecosystem and to distinguish between various communication scenarios and applications.

Empirical results presented in this thesis contribute to the common knowledge about performance of HIP, IPsec and other security components on different classes of resource-constrained platforms operating in lightweight communication systems. We naturally hope that our results will serve as an additional indicator and/or as a "proof of concept" for research communities working on security of 6LoWPAN and the Internet of Things, for industry looking at possibilities to implement HIP-based security products and services and for the HIP community working on migration of HIP specifications from the Experimental to the Standards track in the IETF. An important future work is to standardize lightweight security mechanisms in the context of HIP or similar IP security protocols. This will facilitate their implementation, deployment and usage on resource-constrained devices in the mobile and wireless embedded Internet.

References

[1] HIPL website. [Online] Available at: `http://infrahip.hiit.fi` Accessed 18 May 2010.

[2] Imote2 wireless sensor node platform. Product datasheet. Available online at: `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/Imote2_Datasheet.pdf`. Accessed 18 May 2010.

[3] panOULU network website. [Online] Available at: `http://www.panoulu.net`. Accessed 18 May 2010.

[4] The Symbian Foundation website. [Online] Available at: `http://www.symbian.org`. Accessed 8 June 2010.

[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 18 May 2010.

[6] R. Adler, J. Huang, R. Kong, P. Muse, L. Nachman, R. Shah, C.-Y. Wan, and M. Yarvis. 2007. Edge Processing and Enterprise Integration: Closing the Gap on Deployable Industrial Sensor Networks. In: Proc. of the 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, 2007 (SECON'07), pages 620–630. IEEE.

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

[8] T. Aura, P. Nikander, and J. Leiwo. 2001. DOS-Resistant Authentication with Client Puzzles. In: Revised Papers from the 8th International Workshop on Security Protocols, pages 170–177. Springer-Verlag, London, UK. ISBN 3-540-42566-7.

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

[10] R. Blom. 1985. An optimal class of symmetric key generation systems. In: Proc. of the EUROCRYPT 84 workshop on Advances in cryptology: theory

and application of cryptographic techniques, pages 335–338. Springer-Verlag New York, Inc., New York, NY, USA. ISBN 0-387-16076-0.

[11] C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung. 1993. Perfectly-Secure Key Distribution for Dynamic Conferences. In: CRYPTO '92: Proc. of the 12th Annual International Cryptology Conference on Advances in Cryptology, pages 471–486. Springer-Verlag, London, UK. ISBN 3-540-57340-2.

[12] 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.

[13] 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.

[14] A. Campbell, J. Gomez, S. Kim, C.-Y. Wan, Z. Turanyi, and A. Valko. 2002. Comparison of IP Micromobility Protocols. IEEE Wireless Communications 9, no. 1, page 72.

[15] 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.

[16] Certicom Research. 2009. Standards for Efficient Cryptography. SEC 1: Elliptic Curve Cryptography.

[17] Certicom Research. 2010. Standards for Efficient Cryptography. SEC 2: Recommended Elliptic Curve Domain Parameters.

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

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

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

[21] W. Du, R. Wang, and P. Ning. 2005. An efficient scheme for authenticating public keys in sensor networks. In: MobiHoc '05: Proc. of the 6th ACM international symposium on Mobile ad hoc networking and computing, pages 58–67. ACM, New York, NY, USA. ISBN 1-59593-004-3.

[22] 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.

[23] 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 2010.

[24] 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.

[25] B. Estrem. 2004. Secure Mobile Architecture (SMA) Vision & Architecture. URL `http://www.opengroup.org/products/publications/catalog/e041.htm`. Technical Study E041.

[26] 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`.

[27] 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`.

[28] 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.

[29] 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.

[30] 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 2010.

[31] Forum Nokia. 2009. Nokia Energy Profiler Quick Start Guide. [Online] Available: http://www.forum.nokia.com/Technology_Topics/Application_Quality/Power_Management/Nokia_Energy_Profiler_Quick_Start.xhtml. Accessed 18 May 2010.

[32] 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 2010.

[33] 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 2010.

[34] O. Garcia-Morchon, T. Flack, T. Heer, and K. Wehrle. 2009. Security for Pervasive Medical Sensor Networks. In: MobiQuitous'09: Proc. of the 6th Annual International Conference on Mobile and Ubiquitous Systems. ICST/IEEE.

[35] O. Garcia-Morchon, T. Heer, and K. Wehrle. 2009. Brief announcement: lightweight key agreement and digital certificates for wireless sensor networks. In: PODC '09: Proc. of the 28th ACM symposium on Principles of distributed computing, pages 326–327. ACM, New York, NY, USA. ISBN 978-1-60558-396-9.

[36] 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.

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

[38] A. Gurtov and T. Polishchuk. 2009. Secure Multipath Transport for Legacy Internet Applications. In: Proc. of the Sixth International ICST Conference on Broadband Communications, Networks, and Systems (BROADNETS'09). Madrid, Spain.

[39] D. Halperin, T. S. Heydt-Benjamin, K. Fu, T. Kohno, and W. H. Maisel. 2008. Security and Privacy for Implantable Medical Devices. IEEE Pervasive Computing 7, no. 1, pages 30–39.

[40] 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.

[41] 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.

[42] 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.

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

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

[45] 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.

[46] 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), pages 1–6. IEEE, Dresden, Germany.

[47] 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.

[48] T. Heer and S. Varjonen. 2009. HIP Certificates: draft-ietf-hip-cert-02. Work in progress.

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

[50] T. R. Henderson. 2004. Can SIP use HIP? In: HIP Workshop, 61st IETF meeting.

[51] 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).

[52] 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.

[53] R. Housley. 2005. Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP). IETF RFC 4309. URL `http://tools.ietf.org/html/rfc4309`.

[54] 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 2010.

[55] 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.

[56] 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. Expires in May 2006.

[57] 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.

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

[59] 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`.

[60] 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.

[61] 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.

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

[63] S. Kent. 2005. IP Authentication Header. IETF RFC 4302. URL `http://www.ietf.org/rfc/rfc4302.txt`.

[64] S. Kent. 2005. IP Encapsulating Security Payload (ESP). IETF RFC 4303. URL `http://www.ietf.org/rfc/rfc4303.txt`.

[65] S. Kent and K. Seo. 2005. Security Architecture for the Internet Protocol. IETF RFC 4301. URL `http://www.ietf.org/rfc/rfc4301.txt`.

[66] 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), pages 1–6. IEEE.

[67] A. Khurri, D. Kuptsov, and A. Gurtov. 2010. On Application of Host Identity Protocol in Wireless Sensor Networks. In: Proc. of the 7th IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS'10), pages 358–365. IEEE.

[68] A. Khurri and S. Luukkainen. 2009. Identification of preconditions for an emerging mobile LBS market. Taylor & Francis Group Journal of Location Based Services 3, no. 3, pages 188–209.

[69] 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), pages 1–8. ACM, New York, NY, USA. ISBN 978-1-59593-784-8.

[70] T. Kivinen and H. Tschofenig. 2006. Design of the IKEv2 Mobility and Multihoming (MOBIKE) Protocol. RFC 4621, IETF. URL `http://www.ietf.org/rfc/rfc4621.txt`.

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

[72] 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.

[73] 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.

[74] 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).

[75] 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.

[76] D. Kuptsov, O. Garcia-Morchon, K. Wehrle, and A. Gurtov. 2010. Brief Announcement: Distributed Trust Management and Revocation. In: Proc. of the 29th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC'10), pages 1–3. ACM, New York, NY, USA.

[77] 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).

[78] 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), pages 1–9. IEEE.

[79] N. Kushalnagar, G. Montenegro, and C. Schumacher. 2007. IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. IETF RFC 4919. URL `http://tools.ietf.org/html/rfc4919`.

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

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

[82] 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.

[83] 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.

[84] A. Liu and P. Ning. 2008. TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks. In: IPSN '08: Proc. of the 7th international conference on Information processing in sensor networks, pages 245–256. IEEE Computer Society, Washington, DC, USA. ISBN 978-0-7695-3157-1.

[85] D. Liu and P. Ning. 2003. Establishing pairwise keys in distributed sensor networks. In: CCS '03: Proc. of the 10th ACM conference on Computer and communications security, pages 52–61. ACM, New York, NY, USA. ISBN 1-58113-738-9.

[86] D. Liu and P. Ning. 2003. Location-based pairwise key establishments for static sensor networks. In: SASN'03: Proc. of the 1st ACM workshop on Security of ad hoc and sensor networks, pages 72–82. ACM, New York, NY, USA. ISBN 1-58113-783-4.

[87] 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.

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

[89] 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.

[90] D. McGrew, K. Igoe, and M. Salter. 2010. Fundamental Elliptic Curve Cryptography Algorithms: draft-mcgrew-fundamental-ecc-03.txt. Work in progress. Expires in November 2010.

[91] R. C. Merkle. 1979. Secrecy, authentication, and public key systems. Ph.D. thesis, Stanford University, Dept. of Electrical Engineering.

[92] R. C. Merkle. 1988. A Digital Signature Based on a Conventional Encryption Function. In: CRYPTO '87: Proc. of a Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology, pages 369–378. Springer-Verlag, London, UK. ISBN 3-540-18796-0.

[93] 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.

[94] 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.

[95] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. 2007. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. IETF RFC 4944. URL `http://tools.ietf.org/html/rfc4944`.

[96] R. Moskowitz. 2010. HIP Diet EXchange (DEX): draft-moskowitz-hip-rg-dex-02. Work in progress. Expires in January 2011.

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

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

[99] L. Nachman, J. Huang, J. Shahabdeen, R. Adler, and R. Kling. 2008. IMOTE2: Serious Computation at the Edge. In: IWCMC'08: Proc. of the International Wireless Communications and Mobile Computing Conference 2008, pages 1118–1123.

[100] National Institute of Standards and Technology. 1999. FIPS PUB 46-3: Data Encryption Standard (DES). [Online]. Available at `http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf`. Accessed 18 May 2010.

[101] National Institute of Standards and Technology. 2001. FIPS PUB 197, Advanced Encryption Standard (AES). NIST. [Online]. Available at `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`. Accessed 18 May 2010.

[102] National Institute of Standards and Technology. 2007. NIST SP 800-56A, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography (Revised). NIST. Available at `http://csrc.nist.gov/publications/nistpubs/800-56A/SP800-56A_Revision1_Mar08-2007.pdf`. Accessed 19 Aug 2010.

[103] National Institute of Standards and Technology. 2009. FIPS PUB 186-3: Digital Signature Standard (DSS). NIST. [Online]. Available at `http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf`. Accessed 4 Aug 2010.

[104] National Security Agency. 2009. Suite B Implementer's Guide to NIST SP 800-56A. [Online]. Available at: `http://www.nsa.gov/ia/_files/SuiteB_Implementer_G-113808.pdf`. Accessed 19 Aug 2010.

[105] 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 2010.

[106] National Security Agency. 2010. Suite B Implementer's Guide to FIPS 186-3 (ECDSA). [Online]. Available at: `http://www.nsa.gov/ia/_files/ecdsa.pdf`. Accessed 19 Aug 2010.

[107] J. Nielsen. 1993. Usability Engineering. AP Professional, Boston. ISBN 0-12-518405-0.

[108] P. Nikander, A. Gurtov, and T. Henderson. 2010. Host Identity Protocol (HIP): Connectivity, Mobility, Multi-Homing, Security, and Privacy over IPv4 and IPv6 Networks. IEEE Communications Surveys and Tutorials 12, no. 2, pages 186–204.

[109] 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`.

[110] 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`.

[111] 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.

[112] 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.

[113] 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`.

[114] 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.

[115] 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.

[116] OpenHIP website. [Online] Available at: `http://www.openhip.org` Accessed 18 May 2010.

[117] 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).

[118] R. H. Paine. 2007. Secure Mobile Architecture (SMA) - A way to fix the broken Internet. Inf. Secur. Tech. Rep. 12, no. 2, pages 85–89.

[119] R. H. Paine. 2009. Beyond HIP: The End to Hacking as We Know it. BookSurge Publishing. ISBN 1-4392-5604-7.

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

[121] K. Piotrowski, P. Langendoerfer, and S. Peter. 2006. How public key cryptography influences wireless sensor node lifetime. In: SASN '06: Proc. of the 4th ACM workshop on Security of ad-hoc and sensor networks, pages 169–176. ACM, New York, NY, USA. ISBN 1-59593-554-1.

[122] O. Ponomarev, A. Khurri, and A. Gurtov. 2010. Elliptic Curve Cryptography (ECC) for Host Identity Protocol (HIP). In: Proc. of the Ninth International Conference on Networks (ICN 2010), pages 215–219. IEEE, Menuires, France.

[123] R Development Core Team. 2009. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. [Online]. Available at: `http://www.R-project.org` Accessed 14 June 2010.

[124] J. A. Rice and B. F. Spencer Jr. 2008. Structural health monitoring sensor development for the Imote2 platform. In: Proc. of the SPIE Smart Structures/NDE, pages 1–12. SPIE.

[125] 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`.

[126] R. Roman, C. Alcaraz, and J. Lopez. 2007. A survey of cryptographic primitives and implementations for hardware-constrained sensor network nodes. Mob. Netw. Appl. 12, no. 4, pages 231–244.

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

[128] 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.

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

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

[131] Z. Shelby and C. Bormann. 2009. 6LoWPAN: The Wireless Embedded Internet. Wiley and Sons. ISBN 978-0-470-74799-5.

[132] 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.

[133] 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.

[134] 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.

[135] H. Soliman, C. Catelluccia, K. E. Malki, and L. Bellier. 2008. Hierarchical Mobile IPv6 (HMIPv6) mobility management. IETF RFC 5380.

[136] 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 2010.

[137] K. Sun, A. Liu, R. Xu, P. Ning, and D. Maughan. 2009. Securing network access in wireless sensor networks. In: WiSec '09: Proc. of the 2nd ACM conference on Wireless network security, pages 261–268. ACM, New York, NY, USA. ISBN 978-1-60558-460-7.

[138] 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.

[139] P. Urien. 2009. HIP support for RFID: draft-urien-hip-tag-03.txt. Work in progress. Expires in June 2010.

[140] J. P. Walters, Z. Liang, W. Shi, and V. Chaudhary. 2007. Wireless sensor network security: A survey. In: Book chapter of Security in Distributed, Grid, and Pervasive Computing, Yang Xiao (Eds.), pages 367–403. Taylor and Francis Group.

[141] 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.

[142] 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.

[143] 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.

[144] 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.

[145] 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.

[146] 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.

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

[148] 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.

[149] 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), pages 318–329.

Aalto-DD 2 / 2011

Mobile telecommunication systems increasingly rely on TCP/IP technologies for interconnection of PDAs, smartphones, wireless sensors and other embedded devices. However, only in rare scenarios existing IP security mechanisms can be applied to these systems unmodified. Insufficient computational and battery resources, constraints of wireless networks and varying operational requirements call for protocol optimizations and creation of adaptable security services.

This work evaluates the Host Identity Protocol (HIP) in the context of mobile Internet. HIP splits identifier and locator roles of the IP address, provides cryptographic identities and enables secure mobility. Meanwhile, HIP uses public-key cryptography and can stress lightweight hardware. We identify limitations of standard HIP and analyze possible modifications to its building blocks. The use of alternative security components improves computational and energy efficiency of HIP, making it feasible for constrained environments.

ISBN 978-952-60-4004-2
ISBN 978-952-60-4005-9 (pdf)
ISSN 1799-4934
ISSN 1799-4942 (pdf)

Aalto University
School of Science
Department of Computer Science and Engineering
aalto.fi

BUSINESS + ECONOMY

ART + DESIGN + ARCHITECTURE

SCIENCE + TECHNOLOGY

CROSSOVER

DOCTORAL DISSERTATIONS