Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

No. of dissertation: 01634/2010
Date: 15 February 2010

University of Ljubljana, Faculty of computer and information science issues the following dissertation:

Candidate: **BLAŽ PRIMC**

Title: **AUTHENTICATING IDENTITY ADDRESSING**

Type of dissertation: Undergraduate dissertation

Topic of the dissertation:

A widespread use of digital devices in homes and elsewhere leads to a problem of their interconnectivity. Interconnection of these devices into a network allows a significantly better quality of services. One the challenges that arise when devices are connected is the security. For devices like burglar alarm systems, front door openers, and medical devices, it is absolutely imperative that the access is authenticated and allowed only to legitimate users. In the diploma thesis design a solution for the authentication of identity addressing of devices that are communicating with each other. Use the directions that were developed within the scope of the AuHoNe project as a basis for your work. Design a prototype of key parts of the solution and implement it in Java. The solution should include registration of new devices, resolution of local addresses to IP address, establishment of trust relationship between two networks and resolution of foreign address to IP address. Evaluate the results, describe any limitations, and give directions for possible future work.

Mentor:

Prof. Dušan Kodek, Ph. D.

Dean:

Prof. Franc Solina, Ph D.

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01634/2010
Datum: 15.02.2010

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **BLAŽ PRIMC**

Naslov: **NASLAVLJANJE OVEROVLJENIH IDENTITET**
**AUTHENTICATING IDENTITY ADDRESSING**

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Z vsesplošno razširjenostjo digitalnih naprav v domovih in drugod se pojavlja problem njihove medsebojne povezanosti. S povezavo teh naprav v omrežje je namreč mogoče doseči bistveno kvalitetnejše storitve. Enega od največjih izzivov pri tem povezovanju predstavlja varnost. Za naprave kot so alarmni sistemi, ključavnice in medicinski pripomočki, je absolutno nujno dostop ustrezno overoviti in dovoliti samo uporabnikom z ustreznim dovoljenjem. V diplomskem delu zasnujte rešitev overljivih in naslovljivih identitet, s katero je mogoče preverjati in potrjevati identiteto naprav, ki komunicirajo med seboj. Kot osnovo za razvoj uporabite smernice projekta AutHoNe. V programskem jeziku Java izdelajte prototip ključnih delov rešitve, ki omogoča registracijo novih naprav, razrešitev lokalnega naslova v IP naslov, vzpostavitev zaupanja med dvema omrežjema in razrešitev tujega naslova v IP naslov. Rezultate ovrednotite, opišite morebitne pomanjkljivosti in možne nadaljnje smeri razvoja.

Mentor:

prof. dr. Dušan Kodek

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Franc Solina

UNIVERSITY OF LJUBLJANA
FACULTY OF COMPUTER AND INFORMATION SCIENCE

Blaž Primc

# Authenticating Identity Addressing

UNDERGRADUATE STUDY
DIPLOMA THESIS

Supervisor: prof. dr. Dušan Kodek

Ljubljana, 2010

UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Blaž Primc

# Naslavljanje overovljenih identitet

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU

Mentor: prof. dr. Dušan Kodek

Ljubljana, 2010

*Besedilo je oblikovano z urejevalnikom besedil LaTeX.*

Namesto te strani **vstavite** original izdane teme diplomskega dela s podpisom mentorja in dekana ter žigom fakultete, ki ga diplomant dvigne v študentskem referatu, preden odda izdelek v vezavo!

# IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani/-a **Blaž Primc**,

z vpisno številko **63040136**,

sem avtor/-ica diplomskega dela z naslovom:

**Naslavljanje overovljenih identitet**

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom
  **prof. dr. Dušan Kodek**
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 30.06.2010 Podpis avtorja/-ice:

# Acknowledgements

This work would not have been possible without the help of many people.

I'd like to express my gratitude to **my parents, brother, sister** and **friends** who offered support and advices during my studies.

I'd like to thank to **Prof. Dr.-Ing. Georg Carle** from the University TU München for the invitation to join the team of scientists at The Chair for Network Architectures and Services. Staying there and researching with the team was a great experience. I'd especially like to thank to my supervisors at TUM, **Marc-Oliver Pahl, Holger Kinkelin, Heiko Niedermayer** and **Andres Müller**, for their ideas, guidance and patience through out all the phases of this project. *Danke an Euch alle!*

I'd like to thank to my supervisor **prof. dr. Dušan Kodek** from the University of Ljubljana for assistance during my study years abroad, help with overcoming faculty formalities and general guidance with the thesis.

Last but not least, I'd like to thank to lecturers **Matej Sodin** and **my sister**.

# Contents

# Glossary

**ACMP** Autonomous Control and Management Platform

**ACL** Access Control List

**AES** Advanced Encryption Standard

**AutHoNe** Autonomic Home Networking

**CA** Certificate Authority

**CRAM** Challenge Response Authentication Mechanism

**DHT** Distributed Hash Table

**DNS** Domain Naming System

**DoS** Denial of Service

**HMAC** Hash-based Message Authentication Code

**ISP** Internet Service Provider

**MitM** Man-in-the-middle

**OOB** Out-of-bound

**PDA** Personal Digital Assistant

**PKC** Public Key Cryptography

**PKI** Public Key Infrastructure

**RFC** Request For Comments

**TLD** Top Level Domain

**TTP** Trusted Third Party

# Povzetek

V naših domovih je vedno več elektronskih naprav, ki nam samostojno ponujajo zadovoljivo kakovost storitev. Dejstvo pa je, da se večina naprav ne zaveda drugih naprav in tako predstavljajo bolj ali manj izolirane otoke znanja in sposobnosti. Z medsebojno povezavo vseh teh naprav v domače omrežje, njihovim ustreznim upravljanjem in nadziranjem, bi uporabnikom v njihovih domovih lahko ponudili naprednejše in bolj prilagojene storitve.

Evropski projekt AutHoNe (Autonomic Home Networking) [1] stremi k ustvarjanju inovativnih rešitev za komunikacijo v domačih omrežjih, ki bi omogočale avtonomno upravljanje in nadziranje naprav v domačih omrežjih prihodnosti. V nemškem poddelu projekta AutHoNe, na katedri za mrežne arhitekture in storitve (Lehrstuhl für Netzarchitekturen und Netzdienste) [3] na univerzi TU München, je v razvoju platforma za avtonomno upravljanje in nadzor ACMP (angl. Autonomous Control and Management Platform), [2, 29]. Namen platforme ACMP je povezati naprave v domačem okolju v omrežje s skupnim komunikacijskim standardom in omogočiti usklajeno delovanje naprav, prilagojeno potrebam domačih uporabnikov.

Enega od izzivov za platformo ACMP predstavlja varnost. Ker bo platforma med drugim upravljala tudi naprave kot so alarmni sistemi, ključavnice v vhodnih vratih, itd., mora biti za varnost dobro poskrbljeno. Dostop do sredstev in storitev mora biti glede na aktivnega uporabnika ali omrežno napravo ustrezno kontroliran in nadzorovan. Za zagotavljanje kontroliranega dostopa je nujno, da naprave v omrežju vedo s kom trenutno komunicirajo. Na podlagi tega znanja se lahko sistem za kontroliranje dostopa odloči ali uporabniku oz. napravi dovoli dostop do sredstva oz. storitve ali pa ga zavrne.

Obenem je pomembna tudi zanesljivost platforme, na katero so domači uporabniki obstoječih fiksno ožičenih rešitev navajeni. Izredno pomembno je, da se ukazi namenjeni določeni napravi izvedejo točno tam in ne na katerikoli drugi napravi, ki bi morda pomotoma prejela ukaze. Medtem, ko je pri fiksno ožičenih rešitvah, kjer nastopata le dve napravi, problem rešen s pravilno povezavo naprav, je pri omrežjih, kjer lahko komunikacija poteka hkrati med več napravami potrebna bolj dovršena rešitev. Bistveno pri komunikaciji naprav v omrežju postane, da naprave vedo s kom komunicirajo. S pravilno informacijo o identiteti naprave se lahko izognemo npr. napaki, da bi ukazi za povečanje temperature preko spleta okoliščin končali v hladilniku namesto v napravi za nadzor peči. Oba opisana problema sta rešljiva, če poznamo identiteto aktivnega omrežnega uporabnika ali naprave.

Cilj tega diplomskega dela je zasnovati rešitev overljivih in naslovljivih identitet za potrebe platforme ACMP. Ustrezna rešitev bo omogočila zanesljiv sistem za kontroliran dostop in s tem prispevala k zanesljivost platforme. Pred reševanjem problema smo si

ogledali literaturo, ki se ukvarja s problemoma identifikacije in overitve v računalniških omrežjih. Podroben pregled nad relavantnimi koncepti, uporabljenimi v uveljavljenih obstoječih rešitvah, kot so SNMP, Kerberos, SSH in HIP, nam je skupaj s smernicami projekta AutHoNe in analizo potreb domačih uporabnikov služil za sestavo seznama lastnosti, primerne rešitve za platformo ACMP. Ciljne lastnosti ustrezne rešitve so: avtonomnost domačega omrežja, centralno nadzorovana registracija naprav v domače omrežje, možnost overitve identitete naprav, možnost overitve identitet naprav brez posrednika in v načinu "vsak z vsakim" (angl. peer-to-peer), uporabnost iste identitete v vseh domačih omrežjih, hierarhična naslovna shemo s pripadajočo podporno infrastrukturo, uporabnost in uporabniška prijaznost.

Rešitev temelji na digitalnih certifikatih (angl. Digital Certificate) in infrastrukturi javnih ključev (angl. Public Key Infrastructure). Znotraj vsakega doma ena izmed naprav prevzame vlogo izdajatelja digitalnih potrdil HCS (angl. Home Certificate Service). Naprava HCS ima samo-podpisano (angl. self-signed) digitalno potrdilo in drugim napravam v postopku registracije izda podpisano digitalno potrdilo. Naprava ne more postati del domačega omrežja oz. izkazati pripadnosti domačemu omrežju, če prej ne opravi postopka registracije pri napravi HCS. Izdano in podpisano digitalno potrdilo služi napravi kot identiteta in kot potrditev pripadnosti domačemu omrežju. Z odločitvijo za samopodpisano digitalno potrdilo naprave HCS, kot temelj zaupanja v domačem omrežju, je nadzor nad registracijo naprav v domačem omrežju popolnoma v rokah lastnika omrežja.

Overitev identitete brez posrednika je omogočena z izbiro kriptografije na osnovi javnih ključev. Identiteto naprave je vedno mogoče preveriti samo z digitalnim potrdilom naprave HCS. V postopku preverjanja pristnosti, prisotnost naprave HCS ni potrebna. Obenem pa lahko poljubna naprava overi identiteto poljubne druge naprave v kolikor poseduje digitalno potrdilo izdajatelja identitete, torej naprave HCS. S tem se izognemo potencialni šibki točki, ki bi jo s stališča potrebe po stalni dosegljivosti in možnosti preobremenitve lahko predstavljal strežnik za overitev identitet. Hkrati pa naprave za svoje normalno delovanje ne potrebujejo konstantne povezljivosti z domačim omrežjem (npr. zaradi potreb overitve identitete) in je njihova uporaba lahko bolj svobodna.

Domača omrežja lahko sklenejo tudi medsebojno zaupanje, s katerim se krog naprav ki lahko varno in zanesljivo sodelujejo med seboj znatno poveča. Vzpostavitev medsebojnega zaupanja pomeni, da si omrežji izmenjata digitalni potrdili naprav HCS. Digitalna potrdila HCS naprav služita za overitev identitet naprav iz drugih domačih omrežij, ko npr. gostujoče naprave želijo koristiti sredstva ali storitve našega domačega omrežja. S tem postopkom postane ista identiteta naprave enako uporabna tudi v drugih domačih omrežjih, s čimer je uporabnik razbremenjen večkratnega ustvarjanja identitete za potrebe različnih omrežij.

Za uresničitev hierarhične naslovne sheme, iz digitalnega certifikata naprave in digitalnega certifikata HCS naprave, z uporabo kriptografske zgoščevalne funkcije ustvarimo naslov ACMP. Naslov ACMP je sestavljen iz dveh delov ločenih s piko. Prvi del naslova ACMP predstavlja zgoščena vrednost javnega ključa digitalnega potrdila izbrane naprave. Drugi del predstavlja zgoščena vrednost javnega ključa digitalnega potrdila naprave HCS. Naslov ACMP je globalno unikaten naslov naprave, ki izkazuje identiteto naprave in domačega omrežja.

Ker je naslov ACMP kratek, se lahko uporablja v seznamih namenjenim kontroli

dostopa, za overitev identitete in naslavljanje naprav v vseh domačih omrežjih. Infrastruktura za naslavljanje temelji na sistemu domenskih imen (angl. Domain Name Server). Domenski strežnik poskrbi za pretvorbo naslova ACMP v naslov IP in je operativen, da bi zagotovili avtonomnost omrežja, na napravi HCS v vsakem domačem omrežju. Uvedena je nova vrhovna domena "acmp.", ki je namenjena ločevanju naslovov ACMP od ostalih poizvedb DNS. Iz izbiro strežnika DNS je zagotovljena združljivost naslovov ACMP z vsemi programi, ki znajo koristiti naslove DNS. Z uporabo porazdeljene zgoščevalna tabele (angl. Distributed Hash Table) pa so naprave HCS povezane v "peer-to-peer" omrežje, kar jim omogoča medsebojno sodelovanje npr. za potrebe razrešitve naslova ACMP naprave iz drugega domačega omrežja v naslov IP.

Poleg zasnove zgoraj predstavljene rešitve, je rezultat tega diplomskega dela v programskem jeziku Java realiziran prototip ključnih delov rešitve. V prototipu je trenutno mogoča registracija novih naprav, razrešitev lokalnega naslova ACMP v naslov IP, vzpostavitev zaupanja med dvema domačima omrežjema s poudarkom na varnosti pri izmenjavi digitalnih potrdil naprav HCS in razrešitev tujega naslova ACMP v naslov IP. Na podlagi seznama potrebnih lastnosti rešitve in naknadnih ugotovitev je podana ocena dela, opisane so odkrite pomanjkljivosti zasnove in realizacije. Na koncu je podan še pregled možnosti za nadaljnji razvoj predmeta diplomskega dela.

## Ključne besede:

računalniška omrežja, preverjanje istovetnosti, identiteta, naslavljanje, nadzor dostopa

# Abstract

This thesis tackles with security aspects of future home networks, where all devices will be connected to a home network and controlled by a control and management platform. Due to inclusion of critical devices e.g., alarm system, front door opener, etc., access to devices will have to be strictly controlled. An access control system will be needed by any such platform. Additionally, devices need to ensure that commands are executed at the intended device and nowhere else. Both problems are solvable with reliable identification and authentication of active network users and devices.

The aim of this work is to provide the Autonomous Control and Management Platform (ACMP), which is being developed at the TU München, Chair for Network Architectures and Services [3] with identification and authentication solution for the purposes of access control and reliable device communication. The problem of access control, identification and authentication is analysed, followed by an overview of related work and home environment needs. Based on gathered knowledge a list of system requirements is composed, after which the solution is designed and a prototype displaying key features implemented.

The main contribution of this work are the design and implementation of architecture for authenticating identity addressing. The solution is founded on a home network Certificate Authority (CA) which controls home network membership by issuing certificates to devices. Device certificates are used for authentication and home network membership attestation. The combination of DNS and DHT overlay network is used for hierarchical addressing of device identities. With a mechanism for home network trust establishment, secure and reliable collaboration between devices of different home networks is feasible.

## Key words:

computer networks, authentication, identity, addressing, access control

# Chapter 1

# Introduction

The number of electronic devices installed in our homes is increasing and they are serving us well. But their services could be further improved. Currently devices are mostly unaware of any other devices inside home, creating isolated islands of knowledge and capabilities. By interconnecting these islands, devices could be orchestrated and jointly provide home users with improved services.


Figure 1.1: AutHoNe project logo.

The AutHoNe [1] stands for Autonomic Home Networking and the main objective of the AutHoNe project is to design an innovative home network communication architecture with autonomous components allowing self-managing properties necessary for future home and pervasive scenarios.

Within the German sub-part of the AutHoNe project running at TU München on The Chair for Network Architectures and Services [3], the Autonomous Control and Management Platform (ACMP) [29, 2] is being developed. The platform aims to connect home devices to a network with a common communication and interaction standard and orchestrate their operations according to the user's needs. The ACMP is a distributed platform with software i.e., an agent installed on every device participating in a home network. The agent's capabilities for device control and communication with other agents will allow orchestrated operation adjusted to the needs of home users.

One of the challenges for the envisioned platform is security. As the platform will control and manage most (if not all) devices in homes, including alarm system, front door opener, etc., it will be an interesting target for "digital" burglars. How convenient would it be for burglars, to check with the platform, what valuable objects are in the house, confirm that currently there are no residents present, turn off the alarm system and finally let themselves into the house? Too convenient.

To prevent such scenarios, access to the above mentioned systems should be controlled. An access control system of the platform must ensure that only legitimate users can access sensitive resources. For example, it must deny access to unknown users, allow the postman to deliver mail into our mailbox, let our friends view pictures from our holidays together and give the householder complete control over all devices.

The platform must also provide a high level of reliability that the home users are accustomed to with the current hard-wired devices. Devices connected by a control and management platform need to exchange commands among themselves to provide the state inside homes as required by home users. This might involve sending a command to increase temperature from a room temperature control device to the heater. To transform the user's will into results reliably, the two involved devices need to know who are they talking to. The heater must confirm that the command is indeed coming from the room temperature control device and the room temperature control device must confirm that the receiver of the command is indeed the heater. If devices cannot confirm who are they communicating with, the command to increase temperature could end up at the refrigerator instead of the heater.

## 1.1 Goals of this Thesis

For both problems the solution lies in knowing who the active network user or device is. With reliable knowledge of the user's and device's identity, decisions of an access control system in each device will provide home users with a more secure and reliable platform.

The goal of this work is to provide the devices in the ACMP platform with knowledge of device identities. This knowledge will provide better security through protection of sensitive home network resources using an access control system. Additionally the reliability of the platform will be improved because the devices will be able to reliably communicate with the intended device by confirming the identity of a device and so avoid potential misconfiguration and misuse of devices.

## 1.2 Outline of this Thesis

In Chapter 2 - *Problem Analysis* the problem of access control, identification and authentication in computer networks will be analysed, followed by a review of existing approaches for the problem of identification and authentication. Considering home environment specifics, a list of solution requirements will be composed. Chapter 3 - *Design* will present the design of the solution according to system requirements composed in chapter 2. In Chapter 4 - *Implementation* the prototype is presented along with the development platform and software packages used for its implementation. In Chapter 5 - *Evaluation* the design and implementation will be critically assessed and some ideas for future work will be proposed. The thesis will be concluded with the last Chapter 6 - *Conclusion*, offering final remarks about the outcome of this work.

# Chapter 2

# Problem Analysis

In this chapter the reader is introduced into the thesis problem domain. First the issue of identification and authentication in computer networks is presented. Then a critical overview of related work is presented. Together with home environment requirements, the list of system requirements is composed and described in detail. The chapter is concluded with a summary which outlines the next steps for this work.

## 2.1 Identification and Authentication

Computer systems have resources that need to be protected. The protection of computer resources is achieved through *access control*.

To understand access control, some basic terms are defined:

> **Entity** [39]: "An active part of a system – a person, a set of persons (e.g., some kind of organisation), an automated process, or a set of processes that has a specific set of capabilities."
>
> **Security Policy** [39]: "A set of policy rules (or principles) that direct how a system provides security services to protect sensitive and critical system resources."
>
> **Authorisation** [39]: "An approval that is granted to a system entity to access a system resource."

Examples of an entity are users, programmes, processes, or other computer systems. An example of a system policy can be, a rule that allows users to login to workstations during working hours and deny access outside that time period.

> **Access Control** [39]: "A process by which use of system resources is regulated according to a security policy and is permitted only by authorised entities according to that policy."

A real-life example of an access control is a lock on front doors, allowing entrance only to house residents with keys. In computer systems, an access control can look like a username and a password prompt displayed e.g., when a user tries to login to a workstation

or access a web email account. The aim of access control is to allow access to computer resources only to legitimate entities (i.e., persons, devices, processes, etc.). Legitimate entities are defined in security policy by one or more rules that precisely define properties of legitimate entities (e.g., who, what, when, from where, to where, duration, etc.). The security policy is enforced by the access control.

There are many types of access control. A common access control type is user access control[1]. Each person is given a unique user account representing that person on a computer system. By binding a person to a computer user account, a computer can be configured so that it allows login only to person Alice. By knowing who is using the computer, the user security policy can be enforced by an access control system. Additionally, people can also be held responsible for their actions on a computer system (known as *accountability*[2]).

For user access control and accountability, computer systems need a way to represent people using bit patterns. Furthermore, means for a person are needed to convince computer system that the user account being claimed is indeed his or hers. Two related, yet distinct steps involved in this are *identification* and *authentication*.

> **Identification** [39]: "An act or process that presents an identifier[3] (for instance, a user account name) to a system so that the system can recognise a system entity and distinguish it from other entities."

For example a person can identify itself to the system by entering a user account name, email address, swiping a bank card, smart card, waving a RFID chip, speaking a phrase into a microphone, etc. The identifier is typically considered public information (a user-name field is usually unmasked).

> **Authentication** [47]: "An act or process in which an entity provides additional information (also *authenticator*) that must correspond exactly to the identity professed."

A very common authenticator is a password. Unlike an identifier, the authenticator is considered secret information i.e., password fields are masked, SmartCards are tamper proof, private key in public cryptography must remain secret. Keeping authenticator for an identifier secret and/or safe is critical because the level of authenticator protection directly corresponds to the level of security in the system [47]. For example a user-name and password written to a piece of paper next to a computer's monitor allows anyone to login.

[1] This can be generalised to any entity type, e.g. devices, which are the main focus of our work.

[2] From [39]: "The property of a system or system resource that ensures that the actions of a system entity may be traced uniquely to that entity which can then be held responsible for its actions."

[3] From [39]:"A data object – often, a printable, non-blank character string – that definitively represents a specific identity of a system entity, distinguishing that identity from all others."

There are three basic authentication types (also *authentication factors* [47]):

**Type 1**: "Something you know" - password, PIN, etc.

**Type 2**: "Something you have" - SmartCard, OTP[4] token, USB key, etc.

**Type 3**: "Something you are" - fingerprints, voice prints, retina patterns, etc.

The system is not restricted to the use of a single authentication factor. In an environment with high security needs, two or more authentication factors are required to successfully claim the identity. While multiple authentication factors of the same type provide no increase in security (one password offers about the same security as two passwords), the combination of different authentication factors (also *strong authentication*) strengthens the security of the system. It may be relatively easy to steal a password (type 1) (e.g., reading it from a piece of paper, using social engineering or phishing), but it's harder to steal a token device (type 2), without anyone noticing, and even harder to steal or forge someone's fingerprints or retina patterns (type 3). The benefit of multiple authentication factors is that if only one of the required authentication factors is missing to the attacker, he or she won't be able to authenticate and claim the identity.

While identification and authentication are two distinct processes, they are always combined as a two-step process and are often referred to as just *authentication* [39] which we will use too.

It is important to understand that inexistent or weak authentication methods can render access control ineffective. For instance, source IP authentication is missing in the network layer of the current Internet protocol stack. A typical access control at the network layer is a network firewall which usually enforces access control on IP packet flows based on source IP address specified in the IP packet. However, because the firewall has no means to authenticate the source IP address, it is unable to tell the difference between a genuine and a forged source IP address. Thus an attacker can enter any source IP address, even if he does not own the source IP address, and the IP packet is still valid. The receiving host will accept such packets (they are valid after all) and answer back to the forged source IP specified in the packet. This is known as *IP spoofing* [47] and is frequently used in Denial-of-Service (DoS) attacks.

## 2.2 Related Work

With understanding of authentication and its importance for effective access control, authentication solutions in existing distributed computer platforms are examined. Each related work is briefly introduced and relevant points for this work are exposed and assessed.

### 2.2.1 Simple Network Management Protocol (SNMP)

SNMP is a management framework intended for monitoring and configuration of network devices. In an SNMP managed network, each SNMP managed device runs a software

[4] OTP stands for "one-time password".

called an SNMP agent. The agent has local knowledge of management information (e.g., CPU, memory and network utilisation, system name, network address, location, etc.) which is exposed to SNMP capable clients called managers. The SNMP specifications were developed by the IETF, with the first version appearing in 1988. Currently version 3 is available and is now the recommended version [32], while earlier versions were retired and their usage is not recommended. There are many implementations of the SNMP protocol including open-source, e.g. Net-SNMP [23].

SNMP is relevant for this work because it's a framework for device management similar to the ACMP platform. As such, the SNMP framework also had to solve similar problem, we are addressing in this work - the authentication of devices in a distributed platform. Before SNMP managers can configure a device or gather information about the device, they need to authenticate to the SNMP agents residing on the device. We are interested in how an SNMP agent performs authentication of an SNMP manager.

In SNMP version 1 (SNMPv1), the SNMP managers authenticate themselves to SNMP agents using a *community string*. A community string is actually just another term for a password. When an SNMP managers sends a request for information about device state, the password i.e., a community string is sent along to authenticate the SNMP manager to the SNMP agent. Based on the received community string, the SNMP agent can authenticate the SNMP manager and authorise or deny access to the requested information.

The problem with SNMPv1 and the community strings is the lack of identity authentication between SNMP managers and agents (and vice-versa). The correct community string can only serve for the purposes of authorisation of device state information, but does not provide any information about identity. The SNMP agent and manager cannot confirm the identity of each other using the community string. This makes the community string authentication scheme of the SNMPv1 inappropriate for the ACMP platform, where such functionality is needed. Additionally all communication between SNMP agents and managers, including community strings, is unencrypted. This makes the protocol vulnerable to packet sniffing [47] which allows an attacker to recover the community string and use it to authenticate itself to an SNMP agent.

SNMP version 3 (SNMPv3) introduces the user-based security model [33]. This security model provides the well known user-name and password combination to authenticate SNMP managers to the SNMP agents in order to access sensitive information or functionality. When an SNMP manager requests device state information from an SNMP agent, it sends the user-name and password along to the SNMP agent. With this information the agent can authenticate the identity of SNMP manager and authorise its request accordingly.

While SNMPv3 security mechanisms provide a usable authentication solution, the solution is adapted for the client-server model, where one network device (SNMP manager) authenticates to many others devices (SNMP agents). This is not appropriate for a distributed platform like the ACMP. In a distributed platform like the ACMP, the roles of agents and managers (to use the SNMP terminology) will interchange. In ACMP we anticipate that every device will need to authenticate itself to every other device at some point in time. Using the security model of the SNMP, would mean that each device would need a user-name and password database for all other devices inside a

home network in order to authenticate them. This would not scale well as it would be more and more difficult to maintain such a user-name and password database with the growing number of devices. The security mechanisms used in SNMP are not appropriate for the distributed platform like ACMP. The ACMP needs an authentication solution, appropriate for a network of equals, peers which are capable of peer-to-peer authentication i.e, authenticating each other.

### 2.2.2 Kerberos

Kerberos is a distributed, third-party based, authentication protocol which allows user and server authentication without the need to transmit passwords over an insecure network. The key innovation of Kerberos is that a password can be seen as a shared secret between the user and a trusted third party. Kerberos was developed in the mid-80's as part of MIT's Project Athena. It was released as open-source in 1987 and became an IETF standard (RFC 4120) in 1993. There are several implementations of Kerberos protocol, including the original MIT's [19], another open-source version Heimdal [15] (created to avoid US export regulations) and Microsoft's version of Kerberos[18].

Kerberos is relevant for this work because it provides an authentication solution for computer networks. We are interested how the entity authentication is performed in Kerberos. In contrast to SNMP, the Kerberos authentication solution supports peer-to-peer authentication of entities which is needed by the distributed platform like ACMP. First we will see how Kerberos performs peer-to-peer authentication, followed by the concept of domains of trust and identity grouping which are both applicable for the ACMP usage scenario.

Kerberos introduces two concepts: *key distribution center* (KDC) and *tickets*. KDC acts as a trusted third party (TTP) whose sole purpose is authentication of network users. The role of the TTP is to facilitate interaction between two parties who both trust the TTP [52]. With Kerberos both entities involved in authentication trust the KDC. The KDC holds a user account database (freeing other services of this role) and issues authenticators i.e., tickets to users. Each network participant shares a symmetric secret key with the KDC that ideally only the two know. To authenticate a user to the service, the user first sends a request for the ticket to use a service to the KDC. KDC generates a ticket encrypted with service's secret key and encrypts it again with user's secret key and sends it to the requesting user. A ticket is a credential which a user once it decrypts it, presents to the service as means of authentication. Because only the user knowing the password could decrypt the service ticket, is serves to the service as an authenticator for the user. A more detailed explanation of the protocol can be found in [35, 47].

Unlike the SNMP presented above, the Kerberos supports peer-to-peer authentication, where a user can get authenticated to a service in one scenario and the roles interchange in another scenario. Such an authentication solution is appropriate for a distributed peer-to-peer platform, where all peers have permanent connectivity to the KDC which is always involved in the process of authentication.

Additionally to identity authentication, the KDC or TTP in Kerberos also groups identities into domains. Every Kerberos identity is a member of a Kerberos domain (in Kerberos terminology *realms*). The realm membership is related to the KDC the

entities trust and is revealed during identification (a Kerberos realm is part of an identity) and confirmed with authentication. Kerberos realms solve scalability issues, as well as trust issues. First, scalability issues of single realm and KDC are solved by grouping identities into manageable realms. By limiting the number of identities, the KDCs are not overloaded with authentication requests by clients. Second, separation of identities into realms avoids the need for all entities to trust a single KDC. While trust between different Kerberos realms can be established if needed, this separation represents a default barrier between Kerberos domains of trust. If the two realms trust each other, identities become mobile i.e., can be used in other Kerberos realms. This is very convenient for users because a single user-name and password is usable in all Kerberos realms that trusts their home realm.

The separation of identities into domains applies to the foreseen usage scenario of the ACMP platform. We can imagine two or more home networks. While they could share the Kerberos authentication infrastructure, a power outage in home network where KDC is located would cause problems also in other home networks. A single KDC could quickly become a bottleneck with all entities in many home networks. Additionally convincing all home users to trust a single KDC would probably be very hard and unlikely to happen. So the separation of identities with TTP as it is done with Kerberos carries many advantages for our use case. The Kerberos realm adds autonomous property to a network, a property ACMP platform and this project strive after. The main drawback of Kerberos is the need for a KDC's presence during the authentication. In the case of the ACMP platform some devices will be stationary and will never leave the home, however some devices e.g., laptops, mobile phones will be moved out of the house and might not have connectivity to KDC required for identity authentication. With applications that use authentication often, failing authentication requests would disrupt normal operation on devices outside home network which makes the Kerberos authentication mechanism inappropriate for the ACMP platform. The permanent connectivity to KDC is too strict a requirement. The ACMP can not count on devices be constantly connected to home KDC. The solution capable of offline authentication i.e., authentication without mandatory TTP presence during authentication procedure is needed.

### 2.2.3 Secure Shell (SSH)

SSH is a network protocol which allows secure data exchange between two devices over an insecure network. SSH encrypts all traffic to eliminate eavesdropping, connection hijacking, and other attacks [25]. SSH was developed as a replacement for telnet and other insecure remote shells (rsh, rlogin, rcp) that transmit passwords and other sensitive information unencrypted over a network. The first version (now called SSH-1) of the protocol was developed in 1995 by Tatu Ylönen, a researcher at Helsinki University of Technology, as a response to a password-sniffing attack at his university network [43]. The IETF developed an improved version of the protocol and published it as a SSH-2 standard in 1996. SSH is available for most popular platforms. Two notable implementations of the protocol are open-source OpenSSH [25] and a propriety version by Tectia Corporation [51] founded by Tatu Ylönen.

The interesting part of SSH for this work is the SSH Transport Layer Protocol (SSH-TRANS) [36] which among other security features provides server host authentication

to the client. Host authentication is performed without mandatory TTP (e.g., a KDC in Kerberos) presence in the procedure. To achieve offline authentication the SSH takes advantage of Public Key Cryptography (PKC)[5]. In SSH each server has a pair of cryptographic keys: a private and a public key. During key-exchange in SSH-TRANS protocol, the host's public key and a signature (produced with host's private key) are transmitted to the client which should verify both, to confirm the identity of the SSH server. To enable server authentication, the client needs to confirm the authenticity of the host's public key. SSH specification foresees two trust models:

1. The SSH client has a local database of host-name to public key mappings that can be used to verify the public key received from the server.

2. The SSH client does not have the server's public key, but possesses the CA certificate that issued SSH server's certificate and can verify the server's public key with it.

The RFC [36] lists the pros and cons for each possibility. While the first option requires no centrally administered infrastructure, and no third-party coordination, the maintenance of local host-name to public-key database may become burdensome. Additionally secure synchronisation of local databases across devices in the whole home network would be problematic especially as the network grows. History of hosts file and later implementation of the DNS can serve as a proof. The ACMP platform aims to facilitate usage of devices in home networks and as such must protect home users from tedious and repetitive tasks. Therefore this trust model for authentication is not an option for the ACMP platform.

The alternative is to employ Public Key Infrastructure (PKI) [50, 47] and Certificate Authority (CA) in SSH host authentication. CA takes the role of a TTP to which both entities involved in authentication trust. With PKI, instead of maintaining a local database of a SSH host-name to public-key mapping by each SSH client, (ideally) only a single CA certificate must be stored on SSH clients for the SSH clients to be able to perform host authentication. This significantly decreases the maintenance burden imposed by the previously introduced trust model, where constant synchronisation of SSH clients' databases is required. The drawback is that each server's host public key must be certified by a CA, before host key authentication by the SSH client is possible.

As in the case of Kerberos and a worldwide KDC, one single CA would be possible, but is unlikely. While PKI allows offline authentication i.e., requires no CA presence in the authentication process the authentication request would not flood a central server.

[5]While symmetric encryption algorithms use a single key for encryption and decryption in public key cryptography (also asymmetric encryption) each peer has two mathematically related keys: a private and a public key. One key is used for encryption and the other is used for decryption. Private key must remain secret and the public key can be given to anyone. If Bob wants to send an encrypted message to Alice he uses the public key to encrypt the message. Because only Alice knows the private key, only she is able to decrypt the message. PKC is useful also for digital signatures. Alice can digitally sign a message, certifying she is the author of the message, as follows: Alice writes her message. Using a cryptographic hash function she calculates a hash value of the message. She encrypts the hash value using her private key. The result is a "digital signature" and is sent along with her message to Bob. Bob reads the received digital signature and decrypts it using Alice's public key. The decrypted digital signature is compared to the hash value of Alice's message. If the two match Bob can be sure that the message was signed by Alice's private key which is known only to Alice. See [50, 47] for details.

However, requests for certificate issue could overload the server. The most important issue with single world-wide CA remains standing: the issue of trust. How would a single world-wide entity decide whether an identity for a device should be issued or not? A single, world-wide CA to which everyone would trust is unimaginable that's why many smaller CAs are in use. The implication of many CAs is that whenever a SSH client wants to authenticate SSH servers whose certificates are signed by another CA (e.g., CA-2), it must only securely obtain CA-2's certificate and trust it, by storing it e.g., into "trusted CA system storage" which is accessed by an SSH client upon SSH server authentication.

Several CAs, as in the case of Kerberos, provide separation of entities into groups which is beneficial for our case. With the CA issued and signed certificates for devices, the CA gains control over which devices are member of the group. This is an important feature and very usable for the ACMP platform: the ability to centrally control device membership in a home network. With the PKI trust model, the CA role can be assigned to home network owner and by this he could control which devices are part of his home network.

The PKI solution seems appropriate for our case. However, with the CA acting as a TTP (as in the case of KDC in Kerberos), the CA represents a single point of failure. If CA's private key is compromised by an attacker, all certificates issued by the CA can no longer be trusted. An attacker could, once he acquires CA's private key, issue forged host certificates for his own SSH servers and SSH clients would not be able to distinguish attacker's SSH servers from valid SSH servers. Something similar could happen with the ACMP platform where an attacker could infiltrate rogue devices into home network which would appear friendly but could cause damage. The PKI trust model relies on the fact that CA's private key remains safe.

There is a third option which represents a compromise between usability and security. If the server's host key is unknown to and can not be verified by a SSH client using a CA certificate, the client will usually display the server's host key fingerprint. A host fingerprint is a hash digest of server's host public-key and is *self-certifying* [44]. This property (stems from cryptographic hash functions) allows verification of fingerprint association with a certain public key, without consultation with or an a priori trust to a third party. The client has two options. One option is to bypass authentication of the SSH host key on first connection, taking a chance that connection is under the Man-in-the-middle (MitM) attack. The other (security aware) option is to obtain the host's fingerprint beforehand, using an out-of-band (OOB)[6] mechanism. There are many options for that, including the acquisition of SSH host's fingerprint through a DNS query [37]. Other possible OOB methods include a phone call, an SMS, a piece of paper obtained in person from server administrator, etc. If the client calculated host fingerprint equals to the one obtained using the OOB mechanism, the client can be confident (due to the cryptographic properties of hash functions) that it communicates directly to the intended SSH server. On the other hand, if the host fingerprint doesn't match the calculated one, there could be a MitM attack in progress (the server could also have changed its host key).

This feature is important because in case when the CA certificate is missing and the host-name to public-key mapping is inexistent, this feature enables authentication of the

[6]From [28]: "Out-of-band signalling describes signals that are sent between two parties or two devices that are sent via a path or method different from that of the primary communication between the two parties or devices."

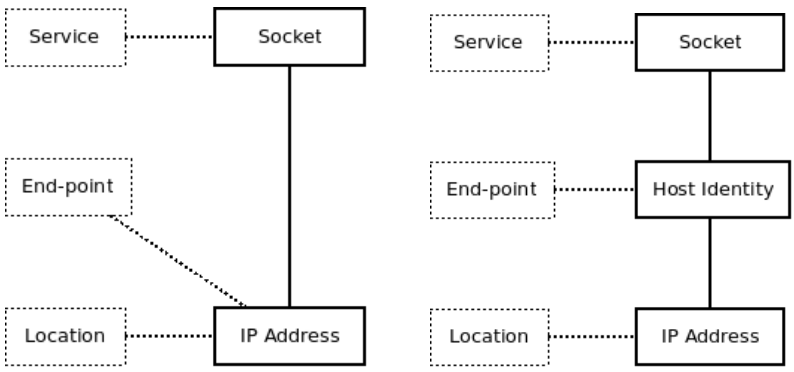
SSH host key, provided the SSH client securely obtained SSH server's host fingerprint. Using infrastructure associating SSH server's DNS name with SSH host fingerprint (as in [37]) the authenticity of SSH server's public-key can be verified without the above mentioned methods. However, it is crucial that the SSH client receives a host fingerprint in a secure manner. This approach for server authentication can be used as a standalone or complementary to the two methods above.

### 2.2.4 Host Identity Protocol (HIP)

HIP aims to provide location independent addressing of network nodes and improved end-to-end host security. It does this by separating the two roles of the IP address [40]:

- The end-point identifier (names the physical network interface in a host) and
- the locator role (topological location of host on the Internet).

HIP introduces a new Host Identity layer between the network and transport layer based on public key cryptography. Upper layer protocols (TCP, UDP, etc.) are bound to HIP addresses instead of an IP address. See figures 2.1a and 2.1b.




(a) Without HIP. (b) With HIP.

Figure 2.1: Protocol stacks.

Introduction of a new layer has several benefits, including support for host mobility, multi-homing, strong authentication between hosts at the TCP/IP stack level, offering protection against MitM and certain DoS attacks [38] and with locally administrated name-space (i.e. self generated public keys), it can provide anonymity. HIP is an experimental effort in the IETF (Internet Engineering Task Force) and IRTF (Internet Research Task Force). Currently all specifications are marked as experimental due to unknown effects that the mechanisms may have on applications and on the Internet at large. Three open-source implementations of HIP exist: HIP for Linux [16], HIP for inter.net [17] from Ericsson and OpenHIP [24] from Boeing.

There are several interesting solutions for the authentication in HIP related to this work. First are the new addresses for the HIP layer, named Host Identity Tag (HIT). A HIT is a hash digest of host's public key, used in protocols to represent the Host Identity

(HI). The HIT is 128 bits long and has the following properties important for this work (full list is available in [40]):

- It is self-certifying (i.e., given a HIT, it is computationally hard to find a Host Identity key that matches the HIT) and
- the probability of HIT collision between two hosts is very low.

While SSH uses SSH fingerprints (which are similar to HITs) only for host authentication, HIP uses HITs also for addressing. With the HIP protocol stack, an IP address is used to locate the host in network and a HIT is used to address a host (by upper layer protocols like TCP, UDP, etc.) and verify its identity. To establish a secure HIP connection both need to be obtained. HIP proposes several resolution mechanisms to translate the DNS name to a HIT and an IP address. Mapping can be done using a configuration file (similar to a hosts file), but more viable options are using the existing DNS infrastructure with a new DNS record [41] or a Distributed Hash Table (DHT)[7] [10].

If a client knows only the target host's IP address, a connection can be established in HIP *opportunistic mode* [8]. This makes the connection prone to MitM attack, similarly to the initial connection using SSH to an unknown host, where no SSH host fingerprint is available.

HIP proposes a flat-structured Host Identity Tag architecture which is simpler and easier to implement [9], however this leads to various problems and limitations. E.g., HITs are only statistically and not globally unique, making a collision highly unlikely, but nonetheless possible [38]. Furthermore, there is a single HIP administrative domain which causes deployment problems, scalability problems and inefficiency of the name resolution system. Additionally flat-structured identifiers are inappropriate for access control aggregation (e.g., network subnets), causing longer access control lists. Due to the above listed reasons, there are HIP extension proposals which add hierarchy support in HIP [12, 9].

HIP also foresees the support for PKI certificate exchange [11] for peer authentication. If the client trusts the CA, CA's signature on host's certificate would prove that host's identity is genuine. While PKI is not always needed, and HIP without TTP protects from MitM attacks, the MitM attacks are hard to prevent, if there is no TTP authentication [40]. TTP could offer protection from MitM attack in opportunistic mode, if the host's key would be issued by a TTP. However, an Internet scale TTP is again an unrealistic scenario.

While HIP provides good security properties, the ACMP software aims to be an add-on to the already existing software on a device. HIP requires either modifications to operating system kernel or a user space daemon to work. With an ACMP platform the focus is on the protection of ACMP traffic and not on the protection of all host traffic. This makes the HIP concept of a new layer a heavy-weight solution for our purposes. For the ACMP platform, an application layer solution will suffice to achieve our goals. With

[7] A DHT is a data structure i.e., a hash table whose contents are distributed over a (large) number of network nodes as opposed to a hash table whose contents are e.g., stored in memory of a single computer. The DHT offers the same functionality as a hash table does. It maps identifying values (keys) to their associated values. The contents of a DHT are stored redundant to avoid the corruption of the DHT structure due to node outfall. For more details see [6].

a secure channel established using ACMP, other channels between applications and hosts can be secured later, if needed.

## 2.3 Home Environment

A home environment consists of devices and their users. Together they form a home network. Trust between members within a home network is higher than trust towards members outside a home network. Devices within the domain usually have the same owner and use the same security settings and as such, enjoy a higher level of trust. Hence, members of a particular home network form a domain. The boundary of a home network should be clear to home users and access to a home network should be tightly controlled. The decision of who or what is to be part of a home network should be in the hands of home users.

While a home network should have clear boundaries and be separated from other home networks, support for the trust relationship between home networks should exist. Information exchange and collaboration between devices from different home networks will allow application designers to develop applications with richer features.

Home networks must be autonomous. The AutHoNe project stands for Autonomic Home Networking and the design of the solutions should strive to achieve autonomy of networks. Firstly, home users are used to that devices within their home are not dependent on an outside entity. That is, everything that is needed for normal operation of a home network must be situated within their home network. An outside entity could present a single point of failure or involve regular service fees. Both are undesirable and should be avoided with solution design. Secondly, most of the management tasks of home network must be done autonomously by the network. This includes also network security which should automatically be provided with as little user interaction as possible.

Technical skills vary a lot between home users, but generally, high level of technical skills cannot be anticipated. Solutions must be designed with such a user in mind and aim to free users from tedious, repetitive tasks.

## 2.4 Requirements to the System

From the analysis of the problem in Section 2.1, followed by the review of the related work in Section 2.2 and specifics of home environments in Section 2.3, the list of system requirements can be composed.

#### R1: Home Network as an Autonomous Domain

Each home network with devices should represent an autonomous domain. The management of device identities must be in control of home users and must be independent from parties outside the home network. Everything needed for authentication solution must be situated inside a home network.

**R2: Central Registration of Devices**

A device in a home network should be responsible for issuing identities. A device cannot become a home network member, unless an identity for the device is issued by a central registration device. This is an important requirement because home users must be in control which devices are part of their home network.

**R3: Provable Device Identities**

For device identification each network device must have a provable identity. A device must be able to attest its home network membership. The identity is the basis for identification of devices and any later access control mechanism based on device identity. This is an important requirement, as a provable identity is the basis for access control mechanisms.

**R4: Offline peer-to-peer Authentication**

Network devices must be capable of authenticating any home network devices without consultation with a TTP. The assumption here is that connectivity to home network will not be always available. Devices must be capable of authenticating any other device inside a home network without consultation with a TTP.

**R5: Mobile Device Identities**

A device identity from one home network must be usable in all home networks. A device must be able to authenticate local home network devices as well as foreign home network devices. This liberates home users from creating a new device identity in each home network and allows collaboration of members from different home networks.

**R6: Hierarchical Addressing Scheme**

A hierarchical addressing scheme with self-certifying properties is needed. The addresses must reflect device membership to a particular home network. The addressing scheme must support the addressing of home network devices as well as foreign home network devices. Naming infrastructure for the addressing scheme must be provided for home network devices.

**R7: Usability and User Friendliness**

Solution should be designed with a home user in mind. The system should provide optimal security for home users with little user interaction.

## 2.5 Summary

This chapter introduced the problem of access control and authentication in computer networks. In Section 2.2 an analysis of existing solutions with advantages and disadvantages per approach presented. The next section presented the specifics of the home environment relevant for our problem. From the two sections, System requirements for the solution were composed and described in detail. None of the existing solutions fulfils all system requirements listed in Section 2.4. The design of architecture for authenticating identity addressing will be presented in Chapter 3 - *Design*.

# Chapter 3

# Design

This chapter presents the design of architecture for authenticating identity addressing according to the system requirements specified in section 2.4. In the first section, architecture and basic concepts are explained. The second section specifies fundamental protocols required by the solution.

## 3.1 Key Infrastructure

### 3.1.1 Home Certificate Service

Digital certificates and Public Key Infrastructure (PKI) are the base for our solution. The root trust in a home network represents a *Home Certificate Service* (HCS) device. HCS owns a self-signed certificate and is a Certificate Authority (CA) for home network. HCS is a central point which controls membership of devices to home network. This fulfils the system requirement R2: *Central Registration of Devices*. Through device registration procedure an unregistered device obtains its certificate from HCS and becomes a member of a home network. The device certificate is signed by the HCS and represents device's identity. The device can authenticate itself using the corresponding private key. The HCS signature of device certificate attests device membership to a particular home network. This fulfils system requirement R3: *Provable Device Identities*.

### 3.1.2 Peer-to-peer offline Authentication

Every home network registered device has a copy of a HCS certificate. With the HCS certificate a device can authenticate all other home network devices whose identities were issued by the same HCS. Devices can be authenticated off-line (without HCS's presence). Devices within a home network are capable of peer-to-peer authentication: any device can authenticate any other device within home network without HCS's presence. This fulfils system requirement R4: *Offline peer-to-peer Authentication*.

### 3.1.3 Home Network Trust Relationship

The selected "one CA per home network" approach effectively separates home networks into domains. Additionally it gives home users total control of identities inside their home network. This supports system requirement R1: *Home Network as an Autonomous Domain*. While home networks are separated per se, trust relationship between home networks can be established, if needed. To establish a trust relationship between home networks, the two home networks need only to exchange HCS certificates and put them into trusted HCS storage. If a home network A trusts home network B, then devices from home network A can verify devices from home network B. With this ability, identities become mobile which is convenient for home users and makes secure collaboration of devices from different home networks possible. This fulfils system requirement R5: *Mobile Device Identities*.

It is critical to ensure the secure trust relationship establishment procedure. The exchange of HCS certificates must be protected, to avoid establishing trust with an attacker's home network. Because home network certificate exchange is likely to occur between two devices in a public space e.g., over insecure wireless channel, the trust relationship establishment protocol should be carefully designed. Design should strive to fulfil system requirement R7: *Usability and User Friendliness*.

### 3.1.4 Hierarchical Addressing

Using a cryptographic hash function over public key in (HCS) device's certificate, self-certifying identities e.g., *deviceID* or *homeID* (from public key in HCS's certificate) are created. By concatenating deviceID and homeID together, hierarchical and globally unique deviceID or *ACMP address* is created. With the naming infrastructure in place, an ACMP address can be mapped e.g., to an IP address. The ACMP address is also short and thus appropriate for ACLs. Aggregation of ACL entries is supported using "*.homeID". Because an ACMP address is derived from a device and the HCS device public key by possessing these two certificates, a device can calculate ACMP address, translate it to an IP address and initiate communication with a device.

For the ACMP address to the IP address mapping naming infrastructure inside a home network (supports system requirement R1: *Home Network as an Autonomous Domain*) is required. Our proposal is a local Domain Name System (DNS) service inside home network, authoritative for ACMP address. A new Top Level Domain (TLD) "acmp." is suffixed to ACMP addresses in order to separate ACMP addresses from other DNS queries. The separation of DNS queries occurs at DNS proxy. ACMP addresses are similar to normal DNS addresses which are already familiar to home users. The DNS also supports Canonical Names (CNAME) records which are used to give user friendly aliases for deviceIDs. Additionally, the advantage of using the DNS naming system is the compatibility of ACMP addresses with existing applications.

To enable collaboration of home networks, all HCS devices are connected to an overlay network where a HCS device can be found if needed. The Distributed Hash Table (DHT) is selected for the overlay because DHTs are decentralised, fault tolerant, scale well and make naming in home networks independent from a third party. DHTs use similar naming space to our deviceIDs, thus HCS devices can use their existing homeIDs as DHT addresses.

With an established trust between home networks, two HCS devices can communicate securely over DHT and establish direct connection between HCS devices. Currently HCS devices collaborate when a foreign ACMP address resolution is required. Because DNS servers are local to each home network, DHT is utilized to contact foreign HCS and forward DNS query to foreign home network DNS server. With this the system requirement R6: *Hierarchical Addressing Scheme* is fulfilled. Additionally the DHT can be utilised to verify that the homeID is indeed globally unique. If an existing homeID is found in a DHT, the HCS device can simply generate a different public key.

## 3.2 Flows

In this section the design for the fundamental protocols: device registration, local and foreign ACMP address resolution and home network trust relationship establishment are presented.

### 3.2.1 Device Registration

Device registration with a HCS is required to provide a new device with a public and private key pair and a HCS signed certificate. Device's IP address is also registered in a DNS server. The device registration has to be initiated at unregistered device. The communication channel between HCS and device is secured (details are explained in Section 3.2.3). The home network administrator has to confirm an incoming request at HCS. If the request is approved, the HCS creates and signs new device certificate and updates the DNS server with information about the new device. Together with the HCS's certificate, the HCS delivers them to the new device. New device stores its certificate and HCS certificate. After successful registration, the new device is now a member of a home network, can be authenticated using a device certificate and addressed using the ACMP address.

The device registration procedure is explained in detail below with a diagram. An unregistered device AlicePDA[1] will register itself towards an AliceAndBobHome HCS device. See Figure 3.1 and diagram explanation below for details.

**Prerequisites**

- ACMP software installed on device AlicePDA and HCS device.
- An HCS device with a configured HCS role.
- IP connectivity between an AlicePDA and the HCS device.

[1]PDA - Personal Digital Assistant: a mobile device also known as palmtop computer.

**Event flow**

1. Alice initiates the device registration procedure on device AlicePDA.

2. Device AlicePDA establishes secure connection to the HCS device.

3. For details see section 3.2.3.

4. Alice enters details about the AlicePDA device. This information includes:

   - Device's friendly name: used as a common name in certificate and as an alias for deviceID in ACMP address.

5. Device AlicePDA sends request details to the HCS device.

6. Home network administrator (this can be Alice) confirms incoming registration request for AlicePDA at HCS.

7. HCS device creates a HCS signed certificate for AlicePDA.

8. HCS device updates the DNS server with:

   - *A record*: where deviceID points to AlicePDA's current IP address. This is used to resolve AlicePDA ACMP address to the IP address.
   - *CNAME*[2] *record*: where device's friendly name points to A record created above. The CNAME record is used to enable a user friendly alias for AlicePDA ACMP address.

9. HCS device sends AlicePDA the following:

   - HCS certificate (used for authentication of other home network devices),
   - AlicePDA certificate signed by HCS device (used as device identity) and
   - AlicePDA private key (used to authenticate the device identity).

10. Device AlicePDA updates its ACMP software configuration with the new identity.

11. Device AlicePDA stores received certificates and private key.

[2]CNAME is short for Canonical Name.

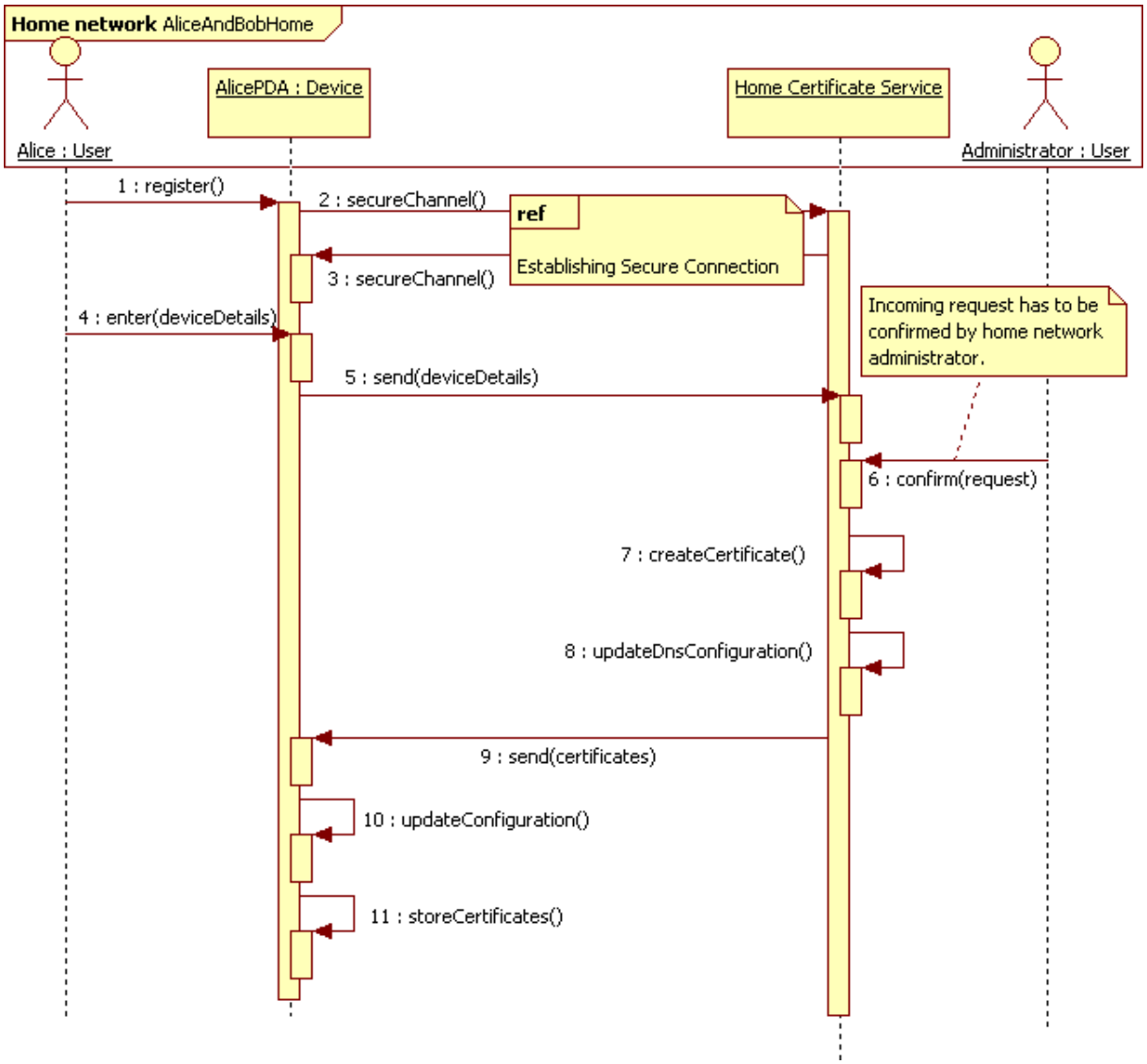



Figure 3.1: Device registration procedure.

### 3.2.2 Local ACMP Address Resolution within Home Network

This flow explains how a device located inside home network resolves a local ACMP address to an IP address. DNS queries are issued to network device where the DNS proxy is running.

A device sends a DNS query to a DNS proxy which decides based on TLD, whether this is an ACMP address or a normal DNS query. In case the TLD equals "acmp."[3], the DNS proxy further inspects the sub-domain name. If the sub-domain name equals local homeID[4], DNS query is forwarded to the local DNS server authoritative for ACMP addresses. The local DNS server answers to a DNS proxy and the DNS proxy forwards the answer back to device that issued the DNS query.

[3]Otherwise the DNS query is sent to preconfigured DNS server for Internet domain names, e.g., ISP's DNS servers.

[4]Otherwise a foreign ACMP address resolution is used. See details in section 3.2.4.

The local ACMP address resolution within the home network procedure is explained in detail below with a diagram. A registered device AlicePDA performs local ACMP address resolution using home network infrastructure. See Figure 3.2 and diagram explanation below for details.

**Prerequisites**

- AlicePDA and BobMobile are registered with home network.
- IP connectivity between AlicePDA and device running DNS proxy.

**Event flow**

1. Device AlicePDA sends DNS query to DNS proxy.
2. DNS proxy inspects incoming DNS query to see, whether this is an ACMP address (procedure continues as step 5) or an Internet domain lookup (procedure continues at next step).
3. DNS proxy forwards DNS query to preconfigured Internet DNS server (e.g., ISP's DNS server).
4. Internet DNS server responds with DNS answer to DNS proxy. Procedure continues at step 8.
5. DNS proxy verifies, if homeID in ACMP address points to local or foreign home network. In the latter case see section 3.2.4 for details.
6. DNS proxy forwards DNS query to local DNS server authoritative for ACMP addresses.
7. Local DNS server responds to DNS proxy with a DNS answer.
8. DNS proxy forwards the DNS answer back to network device AlicePDA.

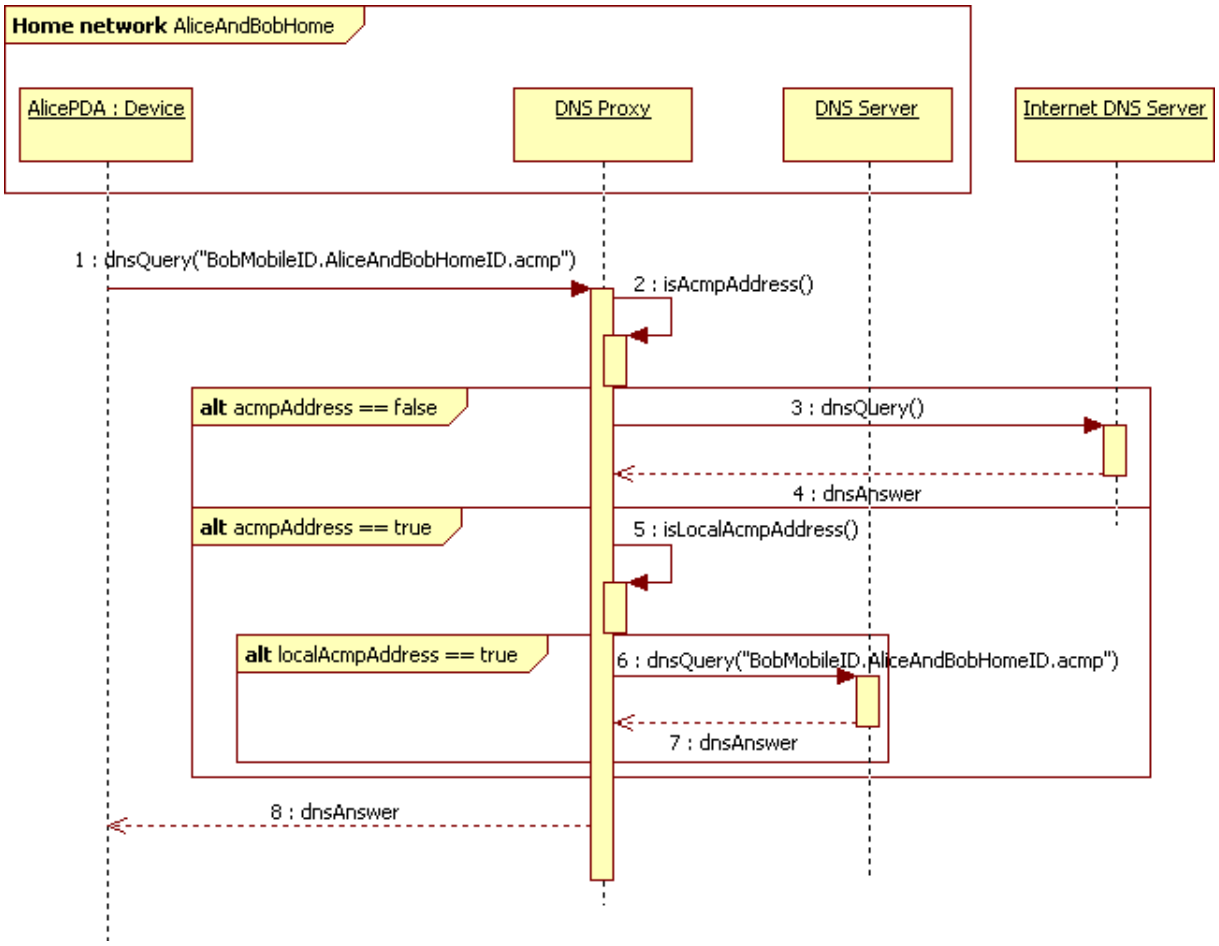



Figure 3.2: Local ACMP address resolution within home network.

### 3.2.3 Home Network Trust Establishment

To establish a home network trust relationship home networks need to exchange HCS certificates. The scenario is the following: two users meet in a public space and want to establish a home network trust relationship. Their devices are registered at two different home networks and are capable of wireless communication. With the exchange of certificates over wireless connection, there is a possibility of a MitM attack. If an attacker manages to hijack the communication channel between devices and relay messages between the two parties, both parties would establish trust with the attacker's home network and not with the intended home network. While trust relationship grants no permissions to foreign home network, an attacker could access all resources the MitM attack victims would authorise to the intended foreign home network later on, not knowing that they have actually authorised access to the attacker.

To prevent such scenarios from happening trust establishment protocol is split into two parts. In first part, the two devices establish a secure communication channel. Using a key-exchange protocol, a secret key is established and used to encrypt the connection between the two devices. In the second part, the two devices exchange certificates over the established secure connection and mutually authenticate each other.

### Establishing Secure Connection

Two devices, with no existing security context, need to establish a secure connection over an insecure channel. With the widespread use of wireless devices, it is a common issue. An example is Bluetooth pairing protocol which is known for its insecurities [45].

To encrypt the connection between devices, a shared secret key between devices needs to be established. To establish a shared secret key, the Diffie-Hellman (DH) key-exchange protocol can be used. DH key-exchange is a cryptographic protocol based on public key cryptography, through which a shared secret key between two parties is established [5, 50]. Even in the case of a passive attacker, where the attacker eavesdrops to the message exchange between parties, the protocol is secure.

The messages in basic DH key-exchange are not authenticated which makes the protocol potentially insecure in the case of an active attacker. If an attacker is capable of intercepting and injecting messages into communication channel, he can perform MitM attack[5] [50]. A MitM on DH key-exchange can be performed as follows. An attacker initiates one DH key-exchange with one party and another key-exchange with the other party. With two established shared keys, the attacker can establish secure channel with both parties and relay messages between the two, making them believe that they are securely communicating directly with one another.

To prevent MitM attacks the DH key-exchange must be authenticated. The purpose of DH session authentication is to confirm that the key-exchange took place with the intended party. Authenticated DH key-exchange is used in ZRTP protocol [13]. The ZRTP protocol is utilized in Zfone[6] for establishment of end-to-end secure channel over the Internet. From [13], the ZRTP "uses ephemeral Diffie-Hellman (DH) with hash commitment, and allows the detection of man-in-the-middle (MitM) attacks by displaying a Short Authentication String (SAS) for the users to read and verbally compare over the phone." SAS string is a hash digest of DH parameters and some other material. Authentication of DH key-exchange gives an attacker only one guess to generate the correct short authentication string (SAS). Therefore a SAS can be quite short [13]: "A 16-bit SAS, for example, provides the attacker only one chance out of 65536 of not being detected".

Our implementation uses a basic DH key-exchange with an additional step at the end of the key-exchange: mutual authentication of peers involved in key-exchange. Authentication is performed by hashing DH parameters and the established secret key. From a hash digest, two short SAS are extracted. One SAS is displayed at one device and the other SAS at the other device's display. While in Zfone the users can ignore displayed SAS values and proceed with telephone conversation even if MitM is present (i.e., SAS values do not match), our protocol requires user interaction before proceeding. Users are required to exchange (e.g., over voice channel) SAS values and enter them into their devices. Devices then perform SAS comparison. If the SAS string entered matches the SAS string calculated by a device, the authenticity of shared secret key is confirmed. If the user mistyped or there is indeed a MitM present during key-exchange, the protocol immediately terminates. Otherwise the protocol continues and the established secret key

[5] MitM attacks are happening also in real-life. See [4].

[6] From [56]: "Zfone is a new secure VoIP phone software product which lets you make encrypted phone calls over the Internet". Zfone's principal designer is Phil Zimmermann, the creator of PGP (Pretty Good Privacy) software.

is expanded and used as keying material for symmetric encryption of data stream.

The first part of trust establishment procedure is explained in detail below with a diagram. Two devices AlicePDA and EveLaptop are involved in this procedure. See Figure 3.3 and diagram explanation below for details.

**Prerequisites**

- ACMP software installed on devices AlicePDA and EveLaptop.
- IP connectivity between the two network devices.

**Event flow**

1. A protocol (e.g., Device Registration as explained in Section 3.2.1) on AlicePDA requires secure communication channel.
2. A protocol on EveLaptop requires secure communication channel.
3. AlicePDA generates DH parameters consisting of:
   - p: prime number and
   - g: primitive root *mod* p.
4. AlicePDA calculates public and private DH key pair.
5. AlicePDA sends DH parameters generated in step 3.
6. EveLaptop calculates its own private and public DH key pair, based on parameters received in step 5.
7. EveLaptop sends its DH public key.
8. AlicePDA sends its DH public key.
9. EveLaptop calculates DH secret key from its DH private key and AlicePDA's public DH key.
10. AlicePDA calculates DH secret key from its DH private key and EveLaptop's public DH key.
11. AlicePDA device calculates and displays SAS A.
12. EveLaptop device calculates and displays SAS B.
13. Alice shares SAS A with Eve over secure channel (e.g., voice).
14. Eve shares SAS B with Alice over secure channel (e.g., voice).
15. Alice enters SAS B into AlicePDA.
16. Eve enters SAS A into EveLaptop.

17. AlicePDA verifies SAS B.

18. EveLaptop verifies SAS A.

19. AlicePDA expands established DH secret key and uses it to encrypt communication channel.

20. EveLaptop expands established DH secret key and uses it to encrypt communication channel.

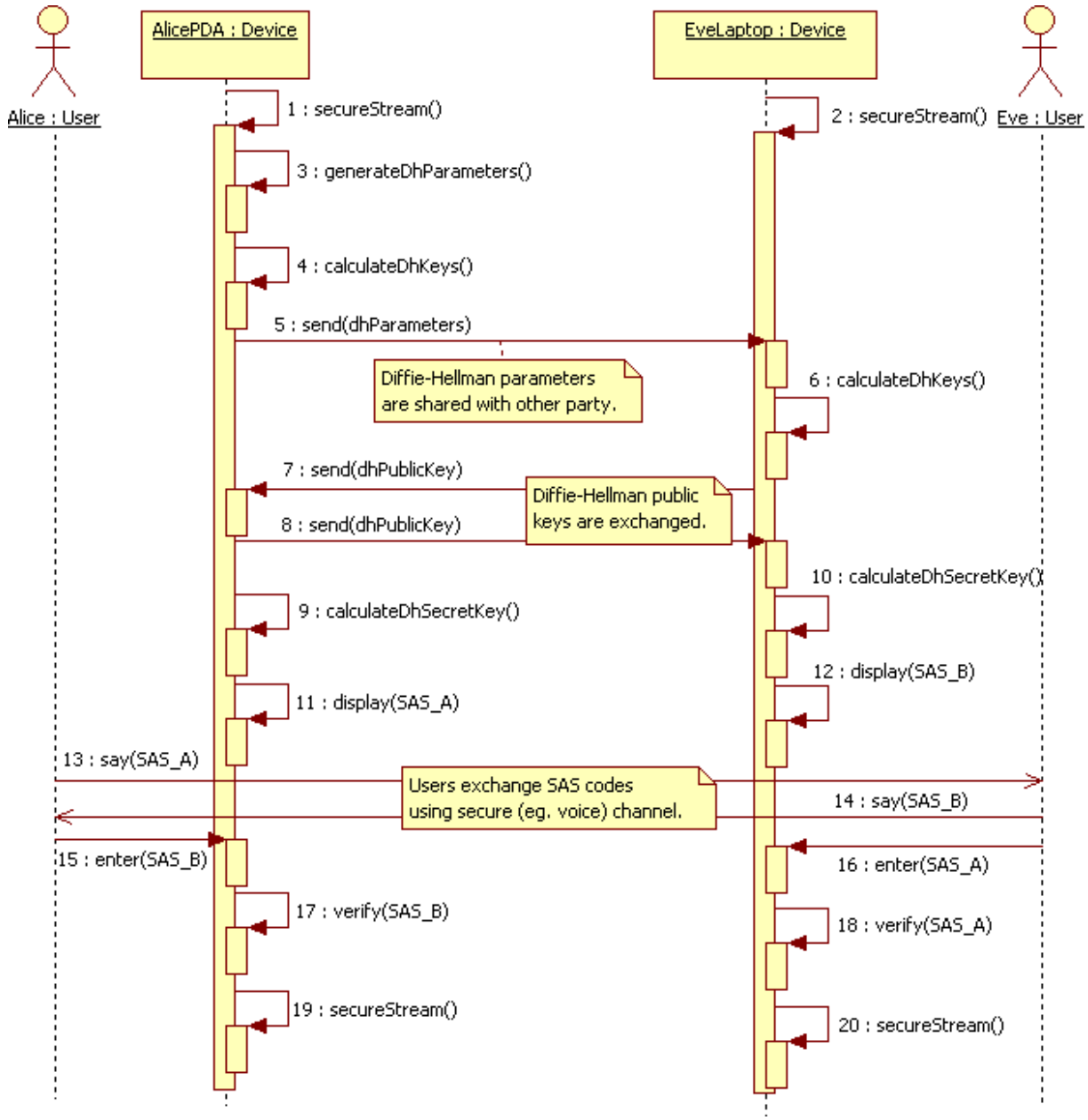



Figure 3.3: Authenticated Diffie-Hellman key-exchange.

### Certificate Exchange and Mutual Authentication

In the previous section the connection between devices was secured. In this part of the protocol, the exchange of certificates takes place. Each device sends its certificate and HCS certificate to other device. In order to confirm the ownership of presented certificates, devices mutually authenticate each other using Challenge-Response Authentication Mechanism (CRAM)[7]. If devices authenticate each other and exchanged certificates are successfully verified, devices store certificates for future use. It is important to note that home network trust establishment does not grant any permissions to foreign network devices. This establishment only lays the groundwork for delegation of permissions and nothing more.

The second part of trust establishment procedure is explained in detail below with a diagram. The two devices AlicePDA and EveLaptop from the first part of trust establishment procedure are involved in this procedure. See Figure 3.4 and diagram explanation below for details.

### Prerequisites

- ACMP software installed on devices AlicePDA and EveLaptop.
- Device AlicePDA is registered at home network AliceAndBobHome.
- Device EveLaptop is registered at home network EveHome.
- IP connectivity between the two network devices.

### Event flow

1. EveLaptop sends certificates to AlicePDA. This includes:
   - EveHome HCS certificate,
   - EveLaptop's certificate.
2. AlicePDA verifies each certificate for:
   - HCS signature,
   - expiration date.

   If verification fails on any of the above steps, the process terminates immediately and all exchanged data is discarded.

[7]Challenge-response authentication mechanism (CRAM): To authenticate party B, party A (or initiator) sends large random number to party B (responder). Party B transforms received number in a special way (e.g., digital signature) and then returns the result to the party A. If the response from party B matches A expectations, the party B has authenticated to party A. See [50] for detailed explanation.

3. AlicePDA sends certificates to EveLaptop. This includes:
   - AliceAndBobHome HCS certificate,
   - AlicePDA's certificate.
4. EveLaptop verifies that each received certificate is valid (similar to step 2).
5. EveLaptop generates random bytes and encrypts them with AlicePDA's public key and sends them to AlicePDA.
6. AlicePDA decrypts received bytes with AlicePDA's private key and calculates message digest of decrypted bytes. The message digest is sent to EveLaptop. If the message digest received from AlicePDA matches the message digest of the bytes sent to AlicePDA, EveLaptop can conclude that AlicePDA possess the private-key for the presented identity.
7. AlicePDA generates random bytes and encrypts them with EveLaptop's public key and sends them to EveLaptop.
8. EveLaptop decrypts received bytes with EveLaptop's private key and calculates message digest of decrypted bytes. The message digest is sent to AlicePDA. If the message digest received from EveLaptop matches the message digest of the bytes sent to EveLaptop, AlicePDA can conclude that EveLaptop possess the private-key for the presented identity.
9. AlicePDA stores certificates for future use.
10. EveLaptop stores certificates for future use.

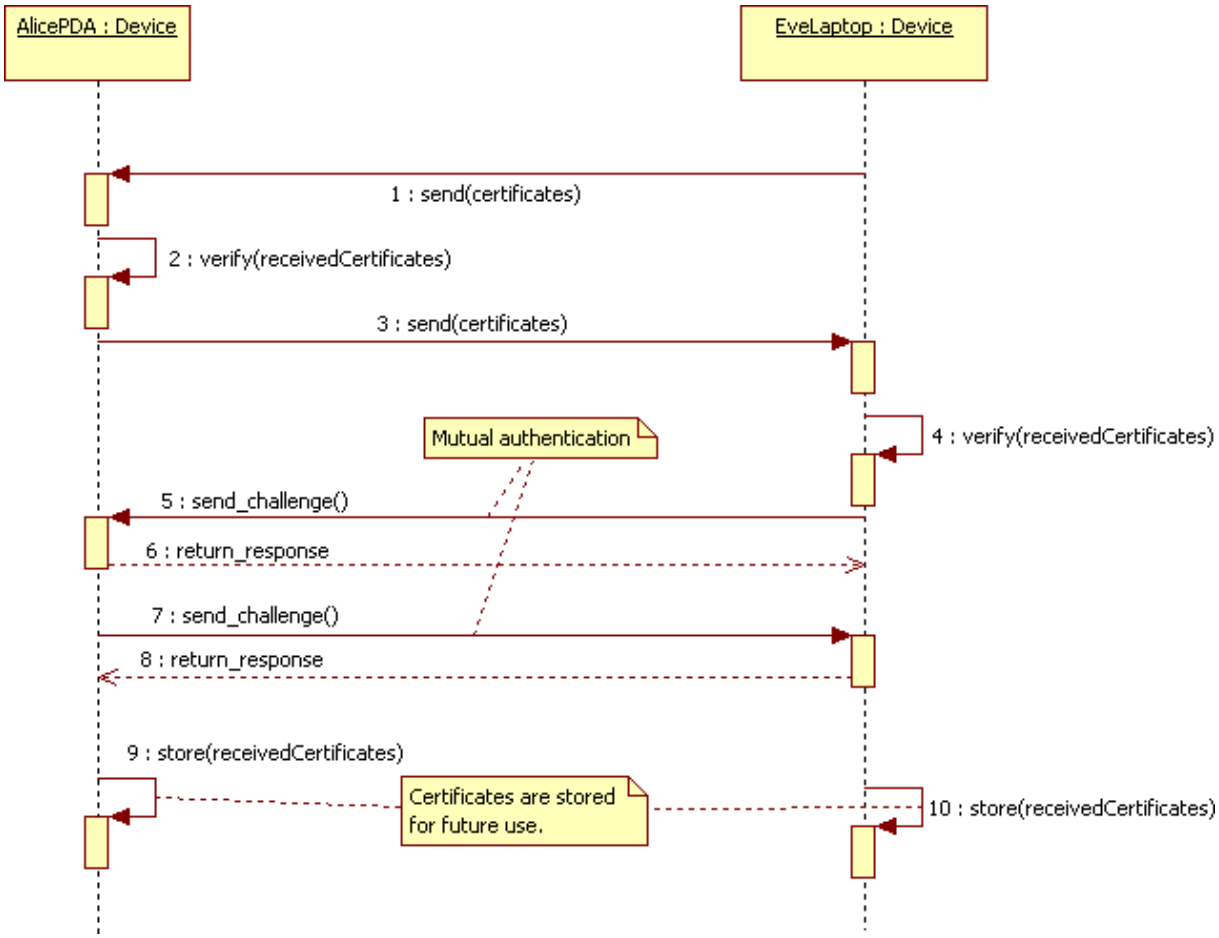



Figure 3.4: Home networks trust establishment procedure.

### 3.2.4 Foreign ACMP Address Resolution within Home Network

This flow explains how a registered device inside home network resolves a foreign ACMP address to an IP address. DNS query is sent to device with a DNS proxy role. The DNS proxy inspects DNS query and concludes that it is an ACMP address (TLD equals "acmp.") for a foreign home network (2nd level domain does not match local homeID). DNS proxy utilizes the DHT client and requests the foreign home network's ACMP service IP address and port. Foreign HCS sends back its HCS service IP address and port inside a DHT reply. Local HCS establishes a direct connection to foreign ACMP service and forwards the DNS query received from local device. Foreign ACMP service forwards the DNS query to its local DNS server and waits for DNS reply. Foreign ACMP service forwards DNS reply back to local HCS and the local HCS forwards DNS reply to device that initiated the DNS query.

The foreign ACMP address resolution within home network procedure is explained in detail below with a diagram. A registered device AlicePDA performs foreign ACMP address resolution using home network infrastructure. See Figure 3.5 and diagram explanation below for details.

**Prerequisites**

- Device AlicePDA is registered at home network AliceAndBobHome.
- Device EveLaptop is registered at home network EveHome.
- Established Trust Relationship between home networks AliceAndBobHome and EveHome. See section 3.2.3 for details.
- AliceAndBobHome and EveHome HCS devices join the same DHT network.

**Event flow**

1. AliceAndBobHome HCS joins a DHT network.
2. EveHome HCS joins the same DHT network.
3. AlicePDA sends a DNS query to DNS proxy. A DNS proxy receives the DNS query from AlicePDA and concludes that it is a foreign ACMP address.
4. A cache is queried for EveHome ACMP service IP address and port number. If cache is valid, steps 5 to 9 are skipped and the procedure continues with step 10. Otherwise a DHT query message is sent.
5. AliceAndBobHome HCS sends a DHT query message[8] addressed to EveHome HCS, requesting EveHome ACMP service IP address and port number.
6. The DHT query message is routed to EveHome HCS.
7. If EveHome HCS receives the DHT query message, it verifies that it has an established trust relation with AliceAndBobHome. Message signature is verified using AliceAndBobHome HCS public key. Keying material in message is decrypted with EveHome HCS private key and stored. A DHT reply message is constructed with EveHome ACMP service IP address and port number. EveHome DHT client adds HMAC to DHT reply message, using keying material provided by AliceAndBobHome.

   **EveHome is not reachable**

   When EveHome HCS is not reachable i.e., DHT request message cannot be delivered to it, the closest DHT node to EveHome responds back to AliceAndBobHome with empty DHT reply message. The procedure continues at step 14.
8. The DHT reply message is routed back to AliceAndBobHome DHT client.

[8] A DHT query message includes keying material encrypted with EveHome HCS public key thus only readable by EveHome HCS device. This keying material is used for authentication of DHT reply message with (with Hash-based Message Authentication Code (HMAC) which assures the integrity and authenticity of message) and to secure direct communication between the two HCS devices later on. A random number (nounce) is added to DHT query message, to avoid replay attacks. Content of the DHT query message is signed with AliceAndBobHome HCS private key.

9. A cache is again queried for a valid entry with an ACMP service IP address and port number of EveHome home network.

   **If cache is invalid, procedure continues with step 14.**

10. AliceAndBobHome HCS establishes a direct connection with EveHome ACMP service. The communication is encrypted. Keying material generated by AliceAndBobHome in step 5 is used to produce encryption keys and initialization vector.

    DNS query that AliceAndBobHome HCS received from AlicePDA is forwarded to EveHome HCS.

11. EveHome HCS forwards received DNS query to (local) DNS server.

12. The DNS reply from EveHome's DNS server is sent back to AliceAndBobHome gateway.

13. AliceAndBobHome HCS receives the DNS reply from EveHome's gateway and forwards it to AlicePDA.

14. This step occurs only if there was no valid cache entry even after a DHT query message was sent. In this case, AliceAndBobHome DNS server responds to AlicePDA's DNS query with an empty DNS response.

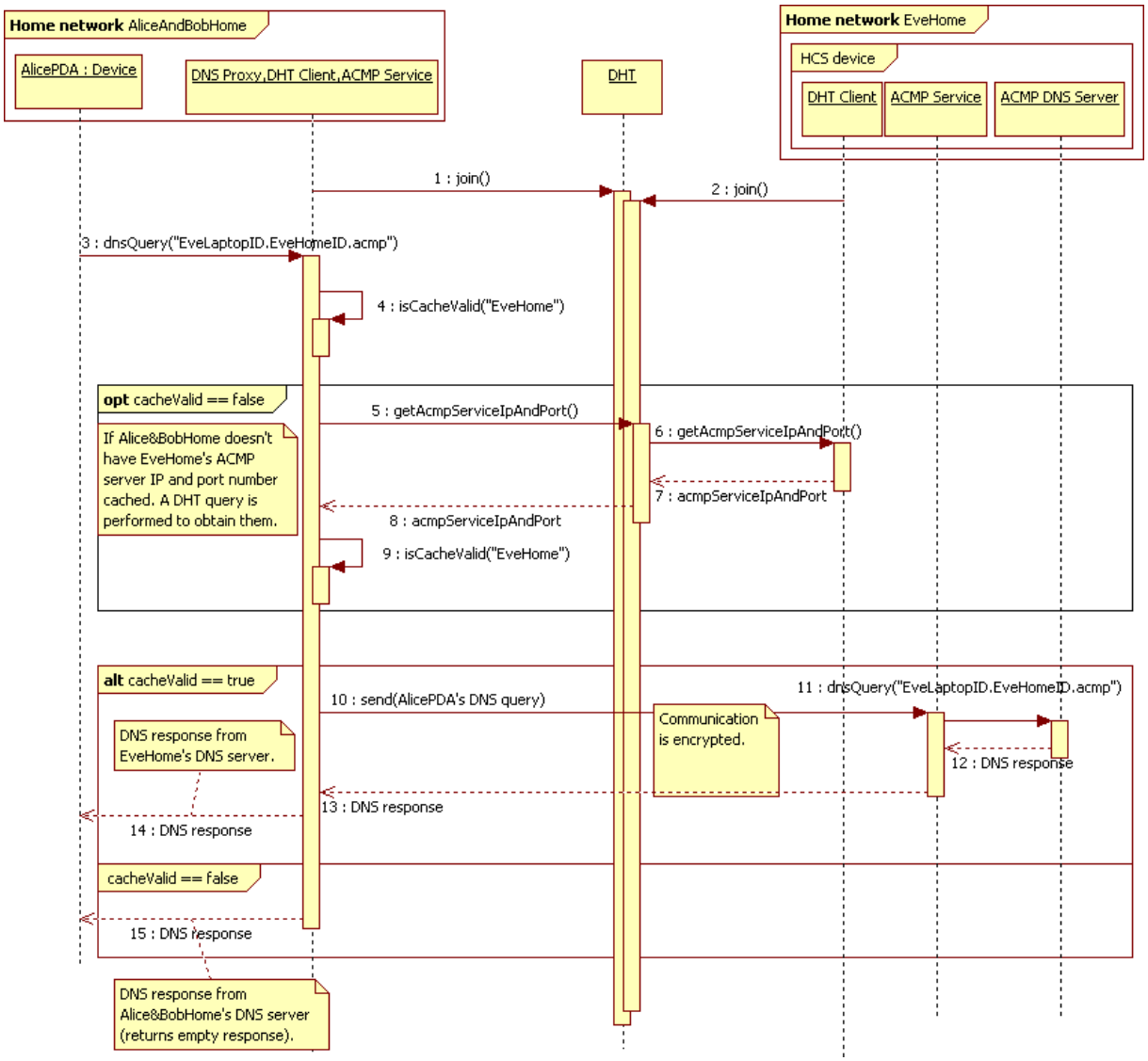



Figure 3.5: Foreign ACMP address resolution within home network.

# Chapter 4

# Implementation

This chapter presents how the proposed solution described in Chapter 3 was implemented. The source code of the prototype is located in folder *src* on the CD accompanying the thesis. Please refer to the Appendix A - *Source Code* for details.

In first two sections the development platform and software packages used for prototype implementation are introduced. The last section introduces the important classes and their methods.

## 4.1 Development Platform

The prototype was developed using a PC computer with an Intel Core 2 Duo 2GHz processor equipped with 4GB DDR2 RAM, running a 32-bit version of Ubuntu [53] v9.10 (Karmic Koala) with Linux kernel v2.6. The prototype was implemented in Java programming language (Sun's Java [49] SDK v1.6) with Eclipse IDE [14] and Subversion [48] for source code version control.

## 4.2 Software Packages

The realization of prototype would not have been possible without the already existing software. This list briefly introduces used software packages and their role in this project.

#### OpenSSL

OpenSSL [26] is an open-source tool-kit implementing the SSL and TLS protocols as well as a full-strength general purpose cryptography library. In this project, the OpenSSL's PKI capabilities are utilized by EasyRSA scripts to generate certificates for devices.

#### EasyRSA

EasyRSA is a set of scripts which interact directly with OpenSSL toolkit and instruct OpenSSL to generate certificates. Scripts are provided by OpenVPN [27] project. In this

project, the EasyRSA scripts are used by HCS to generate HCS and device certificates.

### MyDNS

MyDNS [21] is an implementation of DNS. It is designed to serve DNS records directly from SQL server (MySQL [22] in our case). In this project MyDNS is used to resolve ACMP addresses to IP addresses. See section 3.2.2 for details.

### dnsjava

dnsjava [7] is an implementation of a DNS server in Java. In this project, a subset of dnsjava functionality is used to implement DNS proxy which distinguishes ACMP addresses from a other DNS domain names and forwards the DNS query to the authoritative DNS server. See section 3.2.4 for details.

### FreePastry

FreePastry [30] is an open-source Java implementation of Pastry - a generic, scalable and efficient substrate for peer-to-peer applications. In this project, FreePastry is used to interconnect all HCS devices in a DHT which allows HCS device to communicate with other HCS devices. See section 3.2.4 for details.

### Simple

Simple [46] is a high performance XML serialization and configuration framework for Java. In this project, Simple is used to export and import configuration files of network devices (currently HCS device, device) to XML files.

## 4.3 Important Classes

This section describes main software components of device and HCS prototype implementation. The entire program can be found on the CD accompanying the thesis in folder *src*.

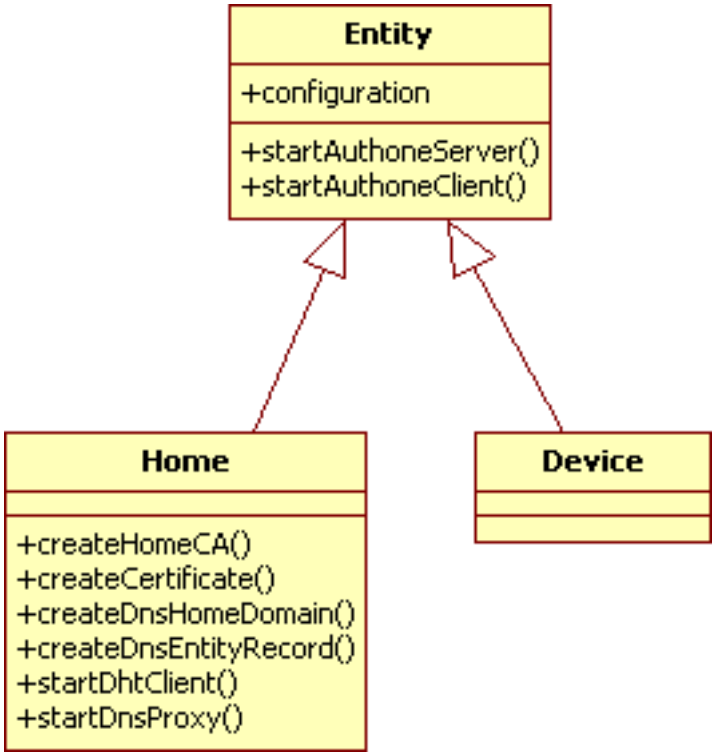



Figure 4.1: Entity and subclasses diagram.

**Entity (package: authone.identities.entity)**

This class represents the base for all entities involved in home network.

- **Entity()**: The call to the Entity class constructor accepts *Configuration* class which reads entity configuration from an XML file.
- **startAuthoneServer(port)**: This method starts an Authone server on *port* specified in *configuration* and awaits Authone client connections.
- **startAuthoneClient(server, port, protocol)**: This method starts an Authone client class which connects to specified server address and port number. Once the connection between client and server is established, the client starts communicating with the server using the specified protocol.

**Home (package: authone.identities.entity)**

This class represents the HCS device and contains functionality needed by HCS.

- **Home()**: The call to the Home class constructor calls the constructor of the super class.
- **createHomeCA(configuration)**: This method creates a self-signed CA certificate for HCS device which is used to sign device certificates. The method executes the *build-ca* shell script included with EasyRSA.
- **createDnsHomeDomain()**: This method updates DNS server with DNS records according to HCS identity. The method is called only when HCS creates brand new CA certificate.
- **createCertificate(name)**: This method creates device certificate signed by HCS certificate. *CommonName* field in certificate is set to *name* which is submitted by device. The method executes the *build-key* shell script included with EasyRSA.
- **createDnsEntityRecord()**: This method updates DNS server with DNS records for new device identity. The method is called after the device certificates are created.
- **startDhtClient()**: This method starts the DHT client. DHT client joins an existing DHT ring if there is a DHT peer specified in configuration file. Otherwise DHT client creates its own DHT ring.
- **startDnsProxy()**: This method starts DNS proxy service on HCS device. DNS proxy listens on standard DNS port number 53 and uses UDP transport protocol.

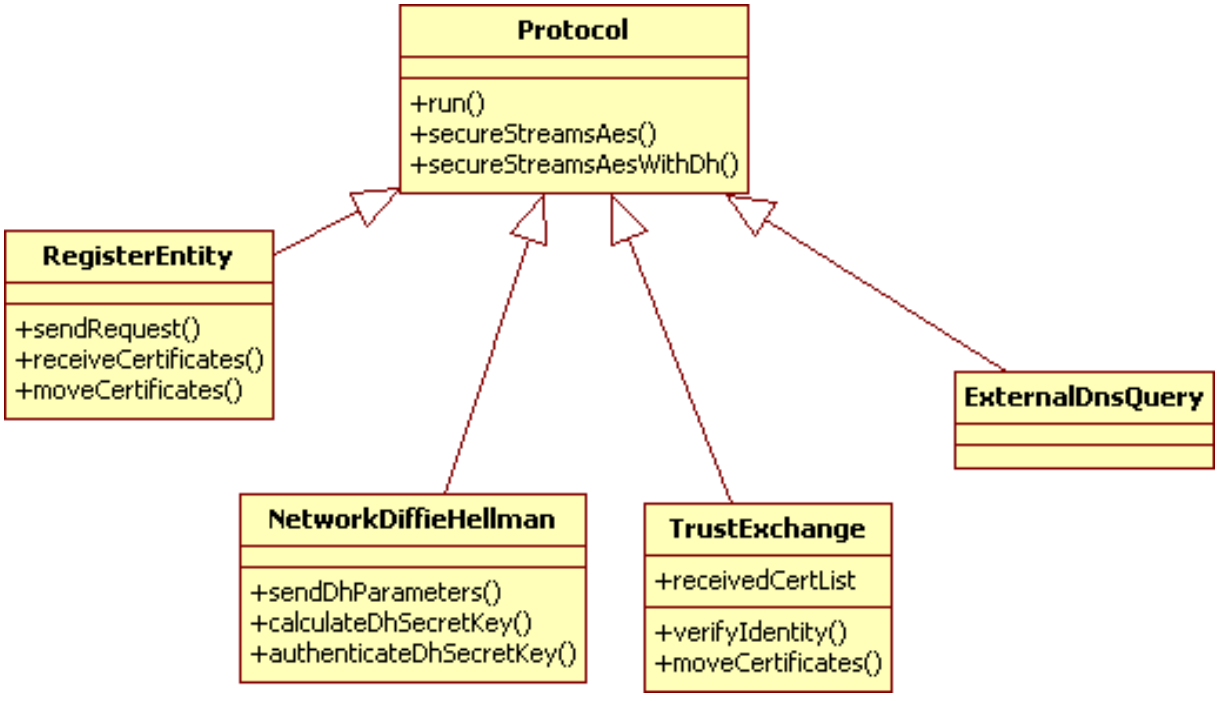



Figure 4.2: Protocol and subclasses diagram.

### Protocol (package: authone.identities.comm.protocol)

This class is the base for all ACMP network protocols used by AuthoneClient and AuthoneServer.

- **Protocol(isServer)**: The call to the Protocol class constructor sets entity role (either server or client) in this protocol.
- **secureStreamWithAes(secretKey)**: This method secures streams with AES encryption. It accepts secret key which is expanded and used for encryption key and initialization vector of the AES encryption engine.
- **secureStreamWithAesWithDh()**: This method secures streams with AES encryption. Firstly the secret key is established and authenticated using *NetworkDiffieHellman* protocol class. The established secret key is passed to the secureStreamWithAes(secretKey) method.

**RegisterEntity (package: authone.identities.comm.protocol)**

This class is the protocol used to register a new device towards HCS.

- **RegisterEntity(isServer)**: The call to the RegisterEntity class constructor calls the constructor of the super class.
- **run()**: This is the main method of RegisterEntity protocol. Depending on peer's role in the protocol, one peer acts as a HCS and the other as unregistered device. For details see Section 3.2.1.
- **sendRequest()**: This method asks the user for device name to appear as *CommonName* field in certificate. User input is sent to HCS.
- **receiveCertificate()**: This method receives certificates created HCS: HCS certificate, device public key and private key pair.
- **moveCertificate()**: This method moves received certificates to permanent storage location.

**NetworkDiffieHellman (package: authone.identities.comm.protocol)**

This class is the protocol for Diffie-Hellman key-exchange over network.

- **NetworkDiffieHellman(isServer)**: The call to the NetworkDiffieHellman class constructor calls the constructor of the super class.
- **calculateDhSecretKey(otherPartyPublicKey)**: In this method, the client sends Diffie-Hellman key-exchange parameters to the server. Both peers exchange their Diffie-Hellman public keys and calculate the secret key.
- **createDhSecretKeyChallenge(part)**: This method creates hash digest of Diffie-Hellman key-exchange parameters and established secret key and displays the specified *part* (or short authentication string) to device display.
- **authenticateDhSecretKey(part)**: This method authenticates the established secret key. It awaits for user input and compares it with the expected authentication string.

**TrustExchange (package: authone.identities.comm.protocol)**

This class is the protocol used to establish Trust Relationship between two home networks.

- **TrustExchange(isServer)**: The call to the TrustExchange class constructor calls the constructor of the super class.
- **run()**: This is the main method of TrustExchange protocol. The protocol involves securing of communication streams, certificate exchange, mutual peer authentication and permanent storage of certificates. For details see Section 3.2.3.
- **verifyIdentity()**: This method creates involves CRAM authentication of peers. The challenge for peer is random data encrypted with public key of the peer being authenticated. The expected response is the hash digest of random data decrypted with peer's private key. The authentication is mutual and must succeed to store peer device and HCS certificates.
- **moveCertificates()**: This method moves received certificates to permanent storage location.

**ExternalDnsQuery (package: authone.identities.comm.protocol)**

This class is the protocol used to transfer DNS query to foreign HCS and receive DNS reply from foreign HCS.

- **ExternalDnsQuery(isServer)**: The call to the ExternalDnsQuery class constructor (used by the server instance) calls the constructor of the super class.
- **ExternalDnsQuery(dnsQuery, remoteHomeId)**: This constructor is called by client instance and takes two input parameters *dnsQuery* and *remoteHomeId*. *dnsQuery* is the raw DNS query packet which will be forwarded to HCS with homeID of *remoteHomeId*.
- **run()**: This is the main method of ExternalDnsQuery protocol. The protocol involves securing of communication streams, forwarding of DNS query and waiting for the DNS reply. For details see Section 3.2.4.

**AuthoneServer and AuthoneClient (package: authone.identities.comm)**

AuthoneServer and AuthoneClient classes are client and server components that handle direct communication between ACMP two peers. They are used by *Entity* class which initiates direct communication between ACMP devices in the following cases:

- Device registration (Section 3.2.1): between HCS and and unregistered device.
- Trust Establishment (Section 3.2.3): between two registered devices from two distinct home networks.
- Foreign ACMP address resolution (Section 3.2.4): between two HCS devices from two distinct home networks.

**DhtClient (package: authone.identities.lookup.local.external)**

This class is a Thread subclass and uses FreePastry library to provide DHT functionality. The class is applicable only for HCS device. The class handles HCS's connectivity to DHT ring. It is used in foreign ACMP address resolution, when the foreign HCS ACMP server's address is unknown or the cached information has expired.

- **run()**: This is the main method of DhtClient class. This method joins an existing DHT ring or starts a new DHT ring.
- **getServiceIpAndPort(remoteHomeId)**: This method starts the search for Authone service IP and port number of the HCS with *remoteHomeId* in the DHT ring.

**DnsProxy (package: authone.identities.lookup.local.external)**

This class is a Thread subclass and uses modified dnsjava code to perform the DNS proxy function. The class is utilized only by HCS devices.

- **run()**: This is the main method of DhtClient class. This method starts a service on a standard DNS port (UDP, port number 53). It inspects each incoming DNS query, to select the target DNS server. The decision is made based on the TLD name in DNS query. In case of a non-ACMP DNS query (i.e. standard domain names), the request is forwarded to predefined DNS server (for instance ISP's DNS). In case of local ACMP DNS name the request is forwarded to local DNS server (see Section 3.2.2). And in the case of foreign ACMP DNS name, the request is forwarded to foreign home network's DNS server (see Section 3.2.4). If the foreign HCS service port and IP address are unknown, the DHT query for foreign HCS device is issued.

# Chapter 5

# Evaluation

This chapter evaluates the solution design and implementation. First the comparison between system requirements specified in section 2.4 and solutions is done. In the next sections limitations of solutions are presented. Last section offers ideas how this work could be improved and continued.

## 5.1 Comparison between Requirements and Solutions

#### R1: Home Network as an Autonomous Domain

With the introduction of HCS CA into home network, the devices are separated into trust domains. By putting CA inside home network, management of identities inside domain is not dependent on or limited by a third party. Identity management of a home network is in the hands of home users. A single domain remains autonomous because all required infrastructure for normal network operation is located inside home network.

#### R2: Central registration of devices

With the introduction of a central device registration procedure presented in Section 3.2.1, home network membership is controlled. Without successful completion of device registration procedure, a device cannot become a member of a home network. Due to security of public key cryptography it is not feasible for a device to falsify its membership of a home network.

#### R3: Provable Device Identities

Each device is given a public and private key pair and a HCS signed certificate. With this credentials a device can authenticate itself and prove its home network membership to any other device which trusts the HCS certificate.

**R4: Offline peer-to-peer Authentication**

The advantage of public key cryptography is that authentication can occur without a CA being present during the authentication process. A device only needs HCS certificate to verify device's home network membership. With public key cryptography offline peer-to-peer authentication is supported.

**R5: Mobile Device Identities**

Device certificate is usable in all home networks, if the two home networks have established Trust Relationship. Once Trust Relationship between home networks is established, a foreign device identity can be verified just as local device identity is.

**R6: Hierarchical Addressing Scheme**

The introduced ACMP address is derived from public key of device identity and has self-certifying property. It is hierarchical, consisting of device identifier (deviceID) and home identifier (homeID). HomeID reflects device home network membership. The addressing scheme and naming infrastructure supports addressing of all home network devices. The naming infrastructure is located inside home network which keeps home networks independent. At the same time potential services fees for normal operation are avoided.

**R7: Usability and User Friendliness**

With the solution design our aim is to provide best security with minimal user interaction required. With the selection of DNS naming system friendly names for devices can be used. Moreover also applications outside ACMP platform can use ACMP addresses. The users are liberated from creating multiple identities. With an established trust relationship to foreign home network, their device identity is usable as it is in home network.

## 5.2 Limitations

**Decentralized Design**

The distributed solution design comes with a price. Firstly all HCS devices are reachable over DHT only if they join the same DHT ring. In other words, in our prototype HCS devices are visible only in their respective DHT ring. Secondly, in order to maintain the structure of DHT even with high churn (rate of node arrival and departure), there is a lot of control traffic between DHT nodes. This could affect low bandwidth DHT nodes.

The decentralized nature of DHT makes DHTs susceptible to attacks which are hard to protect from. A determined and a resourceful attacker has many attack vectors at hand. The most common attacks on DHT are Sybil attack, Eclipse attack and routing and storage attacks [54]. In Sybil attack, an attacker joins DHT with many identities. While this has no ill effects on DHT operation, the Sybil attack is the basis for Eclipse attack. Because of scalability, each node maintains only a few links to other DHT nodes.

If an attacker manages to get his nodes into victim node's routing table, the victim node can be "eclipsed" by malicious nodes. By controlling the traffic from victim node, an attacker could achieve that a specific HCS would not be able to send its traffic to other HCS nodes. This would disrupt or totally disable cooperation between one or more home networks.

The Yu et al. [55] observe that for the proper DHT operation, the number of Sybil nodes compared to the number of honest nodes must be bounded. To limit the number of Sybil nodes, the authors propose a SybilLimit protocol which leverages a key insight on social networks. Because our home network trust relationships form a social network, the proposed solution fits well with the proposed solution and should be further examined.

## 5.3 Future Work

The system design was developed with security in mind, however there are still many possibilities to improve overall security of the system. This section identifies and describes remaining security threats of the current system design and implementation.

### 5.3.1 Device Certificate Identity

#### Theft of Device Identity

Current implementation stores device identity after successful registration on a disk without any additional protection. Device public and private key pair and HCS device certificate are stored as regular files on computer's file-system.

While device certificate is a public part of device's identity and must be distributed to other devices in order to verify its identity, the private key for the identity must remain secret at all times. An attacker could take advantage of physical access to device and simply copy unprotected certificate and private key file from storage device. An attacker could also install malware or trojan horse application which could read the unprotected certificate and private key files and send them to the attacker. Additionally once the certificate and private key are loaded into memory, they can be read by any process, which gives the attacker one additional attack vector.

If an attacker somehow acquires device certificate and private key pair, he is then able to authenticate as that device. Depending on the level of access this device is granted, this can pose lesser or greater risk to device's home network, but also to all foreign networks, where this network device has been given permissions.

The PKI foresees Certificate Revocation Lists for cases digital certificate must be cancelled for some reason. This will be discussed in section 5.3.2. But device identity should be better protected in the first place.

#### Theft of HCS Device Identity

A similar scenario is possible with the current implementation of the HCS device. When the HCS device creates its self-signed CA certificate, the certificate and private key are

stored as regular files on computer's file-system.

As before, the attacker somehow obtains HCS device certificate and private key, either with physical access to HCS device or by installing malware or a trojan horse in the HCS device. HCS identity compromise represents much greater security risk because not only the attacker can now impersonate HCS device, he can also issue new device identities signed with home HCS's private key which cannot be distinguished from identities issued by "real" home HCS device. Moreover, all existing home network trust relationship along with capability to exchange new trusts are also available to attacker.

Whole security model of PKI is broken when the HCS certificate and private key are stolen. As HCS certificate is self-signed, there is no TTP that could revoke a stolen or lost HCS certificate. Once this happens, the only possibility is to discard all issued device certificates and cancel all established home network trusts. Next, all device certificates, including home HCS device's certificate have to be recreated. Finally, all home network trusts must be established again.

The theft of HCS device identity has much greater negative effects than the theft of device identity. Thus the protection of HCS device identity (most importantly its private key) is much more important then the protection of device identity.

#### Protection using TPM

As described, the current prototype stores device certificate and private key as unprotected files on device file-system. One of the possible solutions for certificate protection is Trusted Platform Module (TPM).

The TPM is a secure crypto-processor and features the ability to securely store RSA keys. The private key is only known to TPM and does not have to leave the chip to perform signature and encryption operations. If an application wants to perform signature and encryption operations, TPM requires from the application the key owner password. Without a valid password, the TPM will deny security operations.

Additionally to keys, TPM stores also various key attributes. One attribute defined whether a key can be exported out of TPM or not. If the key is defined as migratable, the key can be exported knowing the key owner's password. If the key is defined as non-migratable, it cannot be exported from TPM chip at all.

In our scenario (HCS) device private key could be imported in secure storage of a TPM. This would protect the key from simple attacks described in Section 5.3.1. Without knowing the key owner's password the attacker could not perform signature or encryption operations. The attacker would also be unable to export the key from TPM, unless he knew the key owner password. If the key is defined as non-migratable the extraction of the key would not be feasible at all.

The work done in [20] focuses on protection of the HCS device key using the TPM module. The main goal of the work, protection of the (HCS) device private key against extraction by malware, was achieved. More details are available in [20].

### 5.3.2 Certificate Revocation

With the current implementation (HCS) device certificates are issued with a preconfigured validity of 10 years. For this period the device can use its certificate without renewing it. However in case the device with a valid device certificate and private key is sold, lost or stolen problems could arise. The new device owner could access all data and services to which the device has access.

The whole platform can be trusted more if device certificates could be revoked on demand. One option would be to issue device certificates with a very short validity. Once the certificate expires, the device certificate is invalid and could not be used to authenticate as device. Usually, a device would renew its certificate just before the certificate would expire. But if a device was stolen, the certificate renewal would be denied by HCS, thus rendering device certificate invalid. The shorter the certificate validity period, the more frequent device must renew its certificate. This could be a potential problem, if device could not connect to HCS for a period of time longer than certificate validity.

Another option is to revoke the certificate at HCS and make this information available to devices that need it. At least two options for PKI exists: Certificate Revocation Lists (CRL) [42] or Online Certificate Status Protocol (OCSP) [31] which should be further examined for suitability.

# Chapter 6

# Conclusion

The aim of this work was to provide an architecture for authenticating identity addressing for the reliable access control mechanisms of the ACMP platform. In Chapter 2 - *Problem Analysis* the problem of authentication in computer networks was analysed. Then a review of related work was performed. Finally, considering home environment specifics, a list of system requirements for the solution was composed. Chapter 3 - *Design* presented the design of the new solution according to the system requirements. Chapter 4 - *Implementation* presented the development platform, used software and important parts of the code. The 5 - *Evaluation* chapter critically assessed the solution design and implementation and proposed some ideas for future work.

The main goal of providing stable fundamentals for access control system in ACMP with the authenticating identity addressing was achieved. We have fulfilled the requirements and successfully implemented a prototype according to our solution design. The prototype is operational, thus confirming our design choices.

A problem with the current decentralised architecture design was pointed out. While home networks with local CA remain a secure domain, the lack of any authority in DHT overlay network, makes it a weak link in the overall strong security design, we have aimed for. To eliminate this threat a possible solution candidate was pointed out.

There are several paths the future research related to this work could take. What our solutions lacks, is a method for certificate revocation. Some possible solutions for this were already discussed. Additionally, the possible replacements for DHT or the options for DHT security improvement should be further researched.

# Appendix A

# Source Code

The program source code is included on the CD accompanying this document. The CD is attached to the back hard cover. The contents of the CD are described next.

**bin/**

This folder contains binary files once the code is compiled. See *INSTALLATION* and *compile.sh* in folder *run/* for instructions on compiling the program.

**config/**

This folder contains HCS device and device configuration files. Each folder represents a configuration for a device. It contains an XML configuration file and sub-folders for device certificates from local home and possibly foreign homes.

**easy-rsa/**

This folder contains EasyRSA scripts which are used by HCS devices to generate device certificates.

**lib/**

This folder contains libraries required by the program.

**run/**

This folder contains Bash shell scripts which can be used for program demonstration.

> **INSTALLATION**: step-by-step instructions to get the code up and running with Debian/Ubuntu Linux OS.
>
> **compile.sh**: compiles source code from src/ directory into bin/ directory.

**start_home1.sh, start_home2.sh**: starts HCS device.

**start_unregistereddevice.sh**: demonstrates device registration procedure.

**start_device1_home1.sh, start_device2_home2.sh**: starts Device1 from Home1 or Device2 from Home2.

**dns_query_to_foreign_home.sh**: demonstrates foreign ACMP address resolution.

**src/**

This folder contains Java source code. Source code is split into packages. Some classes and methods have *javadoc* included.

# List of Figures

# References


[1] (2010) AutHoNe — Autonomic Home Networking. Available at:
http://www.authone.de

[2] (2010) Autonomic Control and Management in Heterogeneous Networks. Available at:
http://pahl.de/

[3] (2010) Chair for Network Architectures and Services, Faculty for Information Science, TU München. Available at:
http://www.net.in.tum.de/en/homepage/

[4] (2010) Contra costa county (San Francisco Chronicle article). Available at:
http://www.sfgate.com/cgi-bin/article.cgi?file=/chronicle/archive/2004/03/18/BAG6S5MUEO1.DTL

[5] W. Diffie, M. E. Hellman, "New Directions in Cryptography", 1976

[6] (2010) Distributed hash table (Wikipedia article). Available at:
http://en.wikipedia.org/w/index.php?title=Distributed_hash_table&oldid=366495441

[7] (2010) dnsjava. Available at:
http://www.xbill.org/dnsjava

[8] (2006) Draft: Establishing Host Identity Protocol Opportunistic Mode with TCP Option. Available at:
http://tools.ietf.org/html/draft-lindqvist-hip-opportunistic-01

[9] (2009) Draft: Extensions of Host Identity Protocol (HIP) with Hierarchical Information. Available at:
http://tools.ietf.org/html/draft-zhang-hip-hierarchical-parameter-00

[10] (2009) Draft: HIP DHT Interface. Available at:
http://tools.ietf.org/html/draft-ahrenholz-hiprg-dht-06

[11] (2010) Draft: HIP Certificates. Available at:
http://tools.ietf.org/html/draft-ietf-hip-cert-03

[12] (2009) Draft: Hierarchical Host Identity Tag Architecture. Available at:
http://tools.ietf.org/html/draft-jiang-hiprg-hhit-arch-03

[13] (2010) Draft: ZRTP: Media Path Key Agreement for Unicast Secure RTP. Available at:
http://tools.ietf.org/html/draft-zimmermann-avt-zrtp-20

[14] (2010) Eclipse IDE. Available at:
http://www.eclipse.org

[15] (2010) Heimdal Kerberos project. Available at:
http://www.h5l.org

[16] (2010) Infrastructure for HIP. Available at:
http://infrahip.hiit.fi

[17] (2010) HIP for inter.net project. Available at:
http://hip4inter.net

[18] (2010) Microsoft Kerberos. Available at:
http://msdn.microsoft.com/en-us/library/aa378747(VS.85).aspx

[19] (2010) MIT Kerberos. Available at:
http://web.mit.edu/Kerberos

[20] S. Mittelberger, *A Certification Service for future Home Networks based on Trusted Computing Technology* B.Sc. Thesis, TU München, Munich, Germany, 2009.

[21] (2010) MyDNS. Available at:
http://mydns.bboy.net

[22] (2010) MySQL. Available at:
http://www.mysql.org

[23] (2010) Net-SNMP. Available at:
http://www.net-snmp.org/

[24] (2010) OpenHIP project. Available at:
http://www.openhip.org

[25] (2010) OpenSSH. Available at:
http://www.openssh.com

[26] (2010) OpenSSL. Available at:
http://www.openssl.org

[27] (2010) OpenVPN. Available at:
http://www.openvpn.net

[28] (2010) out-of-band (Wiktionary article). Available at:
http://en.wiktionary.org/w/index.php?title=out-of-band&oldid=9033911

[29] M.O. Pahl, C. Niedermeier, M. Schuster, A. Müller, G. Carle, "Knowledge-based middleware for future home networks", In IEEE *IFIP Wireless Days Conference* Paris, France, December 2009.

[30] (2010) Pastry - A substrate for peer-to-peer applications. Available at:
http://www.freepastry.org

[31] (1999) RFC 2560: X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. Available at:
http://tools.ietf.org/html/rfc2560

[32] (2002) RFC 3411: An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks. Available at:
http://tools.ietf.org/html/rfc3411

[33] (2002) RFC 3414: User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3). Available at:
http://tools.ietf.org/html/rfc3414

[34] (2003) RFC 3535: Overview of the 2002 IAB Network Management Workshop. Available at:
http://tools.ietf.org/html/rfc3535

[35] (2005) RFC 4120: The Kerberos Network Authentication Service (V5). Available at:
http://tools.ietf.org/html/rfc4120

[36] (2006) RFC 4253: The Secure Shell (SSH) Transport Layer Protocol. Available at:
http://tools.ietf.org/html/rfc4253

[37] (2006) RFC 4255: Using DNS to Securely Publish Secure Shell (SSH) Key Fingerprints. Available at:
http://tools.ietf.org/html/rfc4255

[38] (2006) RFC 4423: Host Identity Protocol (HIP) Architecture. Available at:
http://tools.ietf.org/html/rfc4423

[39] (2007) RFC 4949: Internet Security Glossary, Version 2. Available at:
http://tools.ietf.org/html/rfc4949

[40] (2008) RFC 5201: Host Identity Protocol. Available at:
http://tools.ietf.org/html/rfc5201

[41] (2008) RFC 5205: Host Identity Protocol (HIP) Domain Name System (DNS) Extension. Available at:
http://tools.ietf.org/html/rfc5205

[42] (2008) RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. Available at:
http://tools.ietf.org/html/rfc5280

[43] (2010) Secure Shell (Wikipedia article). Available at:
http://en.wikipedia.org/w/index.php?title=Secure_Shell&oldid=352692231

[44] (2010) Self-certifying identities. Available at:
http://tothink.com/mnemonic/selfcert.html

[45] Y. Shaked, A. Wool, "Cracking the Bluetooth PIN", in Proc. *3rd USENIX/ACM Conf. Mobile Systems, Applications, and Services (MobiSys)*, Washington, USA, June 2005, pp. 39–50.

[46] (2010) Simple - XML Serialization. Available at: `http://simple.sourceforge.net`

[47] J. M. Stewart, E. Tittel, M. Chapple, *CISSP: Certified Information Systems Security Professional Study Guide*. Wiley Publishing, Inc., Indianapolis, 4th Edition, 2008, Ch. 1.

[48] (2010) Subversion. Available at: `http://subversion.apache.org`

[49] (2010) Sun Java. Available at: `http://www.sun.com/java`

[50] A. S. Tanenbaum, *Computer Networks*. Prentice Hall PTR, New Jersey, 2004.

[51] (2010) Tectia Corporation - SSH. Available at: `http://www.ssh.com`

[52] (2010) Trusted Third Party (Wikipedia article). Available at: `http://en.wikipedia.org/w/index.php?title=Trusted_third_party&oldid=338690933`

[53] (2010) Ubuntu project. Available at: `http://www.ubuntu.com`

[54] G. Urdaneta, G. Pierre, M. van Steen, "A Survey of DHT Security Techniques", *ACM Computing Surveys*, 2009.

[55] H. Yu, M. Kaminsky, "SybilLimit: A Near-Optimal Social Network Defense against Sybil Attacks" in Proc. *IEEE Symposium on Security and Privacy* Oakland, USA, May 2008.

[56] (2010) Zfone project. Available at: `http://zfoneproject.com`