Allowing IP Networks to be Securely Renumbered and Shared

Damien Leroy

Thesis submitted in partial fulfillment of the requirements for the Degree of Doctor in Applied Sciences

January 31, 2011

Pole d'Informatique
Information and Communication Technologies, Electronics and Applied Mathematics Research Institute (ICTEAM)
Université catholique de Louvain
Louvain-la-Neuve
Belgium

Thesis Committee:

Gildas **Avoine**	UCL/ICTM/INGI
Olivier **Bonaventure** (Advisor)	UCL/ICTM/INGI
Christophe **De Vleeschouwer** (Chair)	UCL/ICTM/ELEN
François **Koeune**	UCL/ICTM/ELEN
Laurent **Schumacher**	FUNDP, Belgium
Laurent **Toutain**	TELECOM Bretagne, France
Eric **Vyncke**	Cisco Systems, Belgium

Allowing IP Networks to be Securely Renumbered and Shared

by Damien Leroy

Computer Science and Engineering Department
Institute of ICT, Electronics and Applied Mathematics (ICTEAM)
Université catholique de Louvain
Place Sainte-Barbe, 2
1348 Louvain-la-Neuve
Belgium

This work was partially supported by a grant from FRIA (Fonds pour la formation à la Recherche dans l'Industrie et dans l'Agriculture, rue d'Egmont 5 - 1000 Bruxelles, Belgium). This work was also partially supported by the Belgian Walloon Region under its RW-WIST Programme, ALAWN Project.

Acknowledgments

The fulfillment of this thesis would not have been possible without the support — both technical and moral — of many people. First of all, Olivier Bonaventure, my advisor, who hired me four years ago and has guided me throughout this thesis. He defined most of the major directions of my work and redirected me when required. Although I seldom agreed with him and often argued, Olivier has always continued discussing quietly with me. He also spent a lot of his time to promote with me the results of my research for business, which was a time-consuming job far from his preferred activities.

I am grateful to all my coauthors, Mark Manulis, Gregory Detal, Francois Koeune, Julien Cathalo, and the other ALAWN project's members. I have had long and interesting discussions with most of them, and some of their writing finally appear in this document.

I would also like to acknowledge Maoke Chen and Hong Dan, Kenji Masui, Bruno Delcourt, and Alain Fontaine for their help in collecting configuration files for the first part of my thesis. Dominique Margot and Bruno Delcourt have helped us in deploying the SWISH mechanism between the UCLouvain and FUNDP universities.

I am thankful to the members of my thesis committee, Gildas Avoine, Francois Koeune, Eric Vyncke, Laurent Toutain, Laurent Schumacher, and Christophe De Vleeschouwer who took time to review my thesis and to give me some feedback.

My former and current colleagues have helped me, more or less directly and technically, to complete this thesis. The main one is Gregory Detal who has collaborated with me during several months and who has greatly contributed to the SWISH implementation when the C language gave me headaches. A big thanks to him, but also to Sebastien Barre who helped me clarifying some IPv6 or Linux problems, Damien Saucez, our proxy-Google,

who is always open to discussion and able to give relevant references on any topic, Benoit Donnet and Pierre Francois who proof-read several of my papers, and the remaining of the IP Network Lab. Pierre, Cristel and Bruno's theses were also very useful to me as good thesis models.

Finally, there are all my moral supporters that helped me to have some good time here at UCLouvain and during my thesis in general. Not trying to cite them is probably the best way not to forget someone. I just want to thank my girlfriend for her continuous support and for her encouragement required to complete this thesis.

Damien Leroy
January 31, 2011

Contents

Preamble

At the end of 2000 the estimated population of Internet users was about 360 million. Mid-2010 it reached 1.97 billion [Int10]. While the Internet continues to spread, its usage is also more and more present in our everyday's life. In developed countries, people often have access to the Internet at work, at home, and increasingly on mobile devices through 3G or WiFi.

Each of these devices must be assigned an IP address to communicate with others on the Internet. The currently used address space, IPv4, was designed in the 1970s [Pos81]. It has a theoretical capacity of only 3.7 billion addresses whose many are lost due to suboptimal allocation. The exhaustion of the IPv4 addresses is now a matter of months [NRO10]. To address this issue, a new network-layer protocol, IPv6, has been designed [DH98] and implemented. Since IPv6 uses 128-bit addresses, it does not suffer from the same addressing space problem as IPv4.

Unfortunately, IPv6 is also subject to scalability issues, but for other reasons. A key operation to allow IPv6 to scale is to be able to easily change the IP addresses of all devices in a campus or enterprise network. This operation is called *renumbering* [CR96]. Some protocols as *DHCP* [DBV+03] or *Neighbor Discovery* [NNSS07] already solve the problem for end-hosts connected on a LAN. However, these protocols cannot easily be applied to all routers and servers. As automated tools are missing, renumbering a network is often done manually and is thus lengthy and error-prone [BLD05]. The first part of this thesis focuses on secure renumbering and automated tools to help the process.

Besides, more and more mobile phones are equipped with both 3G and WiFi interfaces. In the meantime, the tablet market is growing quickly. These tablets as well as laptops require an internet connection for a large part

of their popular applications. Unfortunately, connecting all these devices to the Internet is still painful.

On the one hand, although 3G has been designed for that purpose, it is still expensive and slow, and a big change is not expected in the next few years. On the other hand, WiFi is usually much faster than 3G and much easier to deploy, mostly indoor. Users have typically access to WiFi at home and at work. Between these locations, connecting to a WiFi network is often difficult and risky. Rogue access points can be set up very easily and more secure solutions are often painful to use. In the second part of this thesis, we present *Secure WIfi SHaring* (SWISH), a protocol that enables to secure WiFi sharing and offers security for both the visited network and the visitor. SWISH is compatible within existing standards and has been deployed in actual networks.

Road map

This thesis is organized in two independent parts. Part I tackles the IP renumbering problem while Part II is about WiFi sharing.

Chapter 1 presents how IPv6 addressing works and why IP renumbering is a key issue for the Internet. Chapter 2 describes a new paradigm for secure address allocation and propagation that can help managing networks that often need to change their IP prefix. Chapter 3 proposes a simple and efficient way to update configuration files in servers and routers by using macros which allows writing configuration that are independent from the IP addresses. Chapter 4 summarizes this part and discusses some perspectives.

In the second part of the thesis, Chapter 5 introduces the issues, mostly in terms of security, of existing WiFi sharing solutions. Chapter 6 details our secure WiFi sharing solution, i.e., its architecture, the protocol and some extensions. Chapter 7 reports on our implementation, on our deployment in campus networks and summarizes the lessons learned from the practical measurements we have achieved. Finally, Chapter 8 concludes on this topic.

Bibliographic Notes

Most of the work presented in this thesis appears in previously published conference proceedings and journals. The list of related publications is shown hereafter:

- Damien Leroy and Olivier Bonaventure. *A secure role-based address allocation and distribution mechanism.* In *Proc. ACM CoNEXT Student workshop*, December 2007.
- Damien Leroy and Olivier Bonaventure. *A Secure Mechanism for Address Block Allocation and Distribution.* In *Proc. IFIP Networking*, May 2008.
- Romain Robert, Mark Manulis, Florence De Villenfagne, Damien Leroy, Julien Jost, Francois Koeune, Caroline Ker, Jean-Marc Dinant, Yves Poullet, Olivier Bonaventure and Jean-Jacques Quisquater. *WiFi Roaming: Legal Implications and Security Constraints.* International Journal of Law and Information Technology, 16(3):205-241, Autumn 2008. Oxford University Press.
- Mark Manulis, Damien Leroy, Francois Koeune, Olivier Bonaventure and Jean-Jacques Quisquater. *Authenticated Wireless Roaming via Tunnels: Making Mobile Guests Feel at Home.* Proceedings of the ACM Symposium on Information, Computer and Communications Security (ASIACCS 2009), pages 92-103, Sydney, Australia, March 2009.
- Damien Leroy and Olivier Bonaventure. *Preparing network configurations for IPv6 renumbering.* International Journal of Network Management, Wiley InterScience, 19(5):415-426, September-October 2009. Online ISSN: 1099-1190, Print ISSN: 1055-7148, DOI: 10.1002/nem.717, Copyright 2009 John Wiley & Sons, Ltd.
- Damien Leroy, Mark Manulis and Olivier Bonaventure. *Enhanced Wireless Roaming Security using Three-Party Authentication and Tunnels.* Proceedings of U-NET'09, Rome, Italy, December 2009.
- Gregory Detal, Damien Leroy and Olivier Bonaventure. An Adaptive Three-Party Accounting Protocol. CoNEXT Student Workshop '09, pages 3-4, December 2009. Rome, Italy.

- Damien Leroy, Gregory Detal, Julien Cathalo, Mark Manulis, François Koeune and Olivier Bonaventure. *SWISH: Secure WiFi Sharing.* Computer Networks, Special Issue on "Network Convergence", 2011.

Part I

Renumbering in IPv6 Networks

Chapter 1

Introduction

The current Internet Protocol (v4) was designed in the 1970s. At that time, IP addresses were divided in classes, and organizations willing to connect to the Internet had to obtain an address block from the IANA. In the late 80s, the class-based addresses combined with the growth of the Internet caused two main problems [FLYV93]. First, the sizes of class-A, class-B and class-C addresses were too rigid. Second, projections in the early 90s indicated that the 32 bits Internet address space would become an increasingly limiting resource. The first problem was solved by the introduction of Classless Interdomain Routing (CIDR) [FLYV93] that supports variable size subnets. To face the second problem, the development of IP next generation, now known as IPv6, started [BM95]. Compared to IPv4, the main benefit of IPv6 is its 128-bit address space.

From a scalability viewpoint, a key element of an addressing architecture is how address blocks are allocated. With IPv4, address blocks were initially allocated in a first-come first-served basis. Later, two types of address blocks were defined : *Provider Independent* (PI) and *Provider Aggregatable* (PA). PI address blocks are assigned by routing registries to ISPs and large customers only. PA address blocks are assigned by ISPs, from their PA block, to smaller customers. When a PI address block is allocated to an organization, this organization can use this address block while being connected to any provider. On the other hand, if it receives a PA address block from provider X, it cannot switch to provider Y without renumbering, i.e., updating all its IP addresses.

With IPv6, the initial address allocation plans [RL95] were strongly in favor of mainly allocating PA address blocks to avoid overloading the Internet routing tables, and thus to permit routing scalability. The first policies used by Regional Internet Registries (RIR) basically assumed that IPv6 addresses would only be allocated to ISPs [CSM+09]. Today, end-user organizations are lobbying to force the RIRs to also allocate PI address blocks to them [Pal09]. Their main motivation is that, unfortunately, practical experience has shown that it is very difficult for a customer network to renumber when it needs to change provider. Despite the widespread use of automatic configuration of end-hosts, the addresses used by the routers are mostly manually configured. Moreover, renumbering requires being able to update all configuration files in which IP addresses appear, which include DNS configurations, firewall access lists, DHCP configurations, etc [BLD05].

Besides, it can be considered that an IPv6 site will have to renumber one or several times during its existence. This can be a consequence of a network growth or merge, or, as claimed earlier, due to a provider change. From this perspective, the switch to IPv6 can be seen as an opportunity to design the network addressing in a way that makes further renumbering easier.

In this part of the thesis, we present methods and tools that can be used to ease the renumbering of IPv6 networks. We predominantly discuss renumbering steps, namely, the address block allocation between entities, the announcement of new prefixes through a network and the update of host configurations to take new prefixes into account.

The remainder of this chapter is organized as follows. Sections 1.1 and 1.2 present the state of the art around IPv6 addressing and can be skipped by readers familiar with this topic. Section 1.3 explains the renumbering issue in more details. Finally, this chapter concludes with the road map of this part of the thesis.

1.1 IPv6 Address Format

Compared to IPv4, the main benefit of IPv6 is its 128-bit address space. The size of this address space fully solves — at least in theory — the IPv4

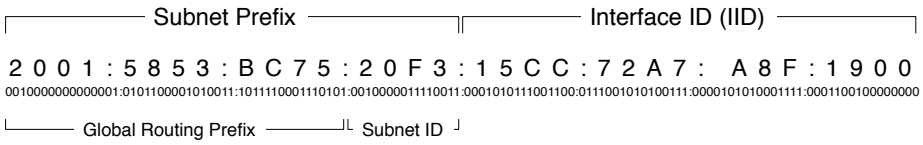


Figure 1.1: Format of a unicast IPv6 address

address exhaustion issue[1]. Additionally, it permits to improve considerably the subdivision and structure of the address space in the same way it is easier to organize the phone numbers by country and by region even if all the countries do not have the same number of subscribers.

The IETF standards [HD06] define the format of an IPv6 address, which is represented at Figure 1.1. An IPv6 address is typically represented as a hexadecimal string and more precisely by a set of eight 4-hexadecimal-digit fields[2], separated by colons. It is not necessary to write the leading zeros in an individual field, but there must be at least one numeral in every field. Only unicast IPv6 addresses will be useful to understand this chapter, so the format of any- and multicast addresses will not be discussed here. These unicast addresses are made of two part of 64 bits.

The leftmost 64 bits of an unicast IPv6 address (`2001:5853:BC75:20F3` in our example) corresponds to the *subnet prefix*. A subnet prefix identifies an IP subnet, i.e., a local link where hosts are interconnected with hubs or switches. A prefix will always be written giving a full IP address filled with zeros for out-of-prefix bits, and its size in bits. In our example, it could be written `2001:5853:BC75:20F3:0:0:0:0/36`. For more convenience, several zeros of an IPv6 address can be replaced by two colons. As a result, the subnet prefix of our example would be written `2001:5853:BC75:20F3::/64`.

The rightmost 64 bits of an IPv6 address are called the *Interface Identifier* (IID) and are used to identify a specific network interface connected on a subnet. For obvious reasons, this IID must be unique on a subnet and for this purpose, it was first proposed to compute these 64 bits based on Ether-

[1] A 128-bit address space corresponds to $3.4 \cdot 10^{38}$ addresses, much more than atoms on Earth. IPv4 address space has $4.2 \cdot 10^{9}$ which is lower than the number of inhabitants on Earth.

[2] In this chapter, fields in addresses are given a specific name, for example, *subnet*. When this name is used with the term *ID* as an identifier after the name (e.g., *subnet ID*), it refers to the contents of the named field. When it is used with the term *prefix* (e.g., *subnet prefix*), it refers to the address from the left up to and including this field.

net MAC address [IEE97]. Mainly for privacy reasons, other way to generate these IID, either randomly [NDK07] or using cryptography [Aur05], have also been proposed.

The subnet prefixes are not directly allocated by the Regional Internet Registries (RIR)[3] to the subnet. Instead, an edge network (i.e., a company, a university, etc) will typically receive from its ISP a shorter prefix that the network will be in charge to allocate in its own network. In the given example, the global routing prefix received by the edge network is `2001:5853:BC75::/48`. The subnet ID, `20F3`, has been locally allocated by the network within its infrastructure.

1.1.1 Global versus Local Addressing

IPv6 enables both global (i.e., routable through the Internet) and local addressing. The easiest way to recognize them is that the `2000::/3` prefix is reserved for unicast global addressing whereas the `FD00::/8` prefix is for local addressing. The local prefix used by an edge network is self-assigned and should be chosen to avoid collision with local networks of other organizations. Such local addresses are called Unique Local Addresses (ULA) [HH05].

Link-local addresses are also allowed in IPv6. They use a fixed subnet (`FE80::/64`) and must be confined on a subnet. These addresses will not be used in our work but had to be mentioned as they are often used.

1.2 IPv6 Address Allocation

The address allocation architecture is a key element for the scalability of the Internet. In this section, the way IPv6 addresses are allocated today to ISPs, edge-networks, LANs and hosts is explained.

[3] RIRs manage, distribute and register public Internet Number Resources. There are currently five RIRs: AfricNIC, APNIC, ARIN, LACNIC, RIPE NCC.

1.2.1 Address Block Allocation to Internet Service Providers (ISP) and Edge Networks

The main consequence of using *Provider Independent* (PI) addresses is that each of these address blocks appear in the BGP routing tables used in the default-free zone[4]. This would lead to scalability issues, mainly if we consider the large address space available with IPv6.

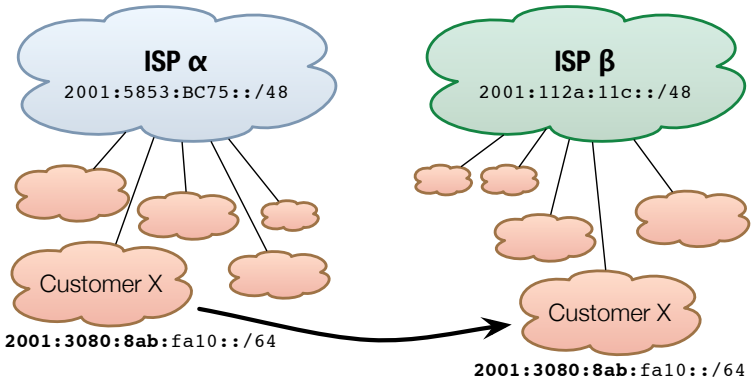
On the contrary, since *Provider Aggregatable* (PA) addresses are chosen as a subset of the ISP address block, there is no need for any extra-route out from ISP. However, as depicted on Figure 1.2, a network using PA and willing to change its ISP will obtain a new address block if this change occurs.

In practice, IANA allocates IP address blocks to the RIRs for further allocation to their members. The hierarchical management structure, shown on Figure 1.3, is that IANA allocates IP addresses to Regional Internet Registries (RIRs), who in turn allocate address space to large ISPs and Local Internet Registries (LIRs) within their respective regions, who then assign them to smaller ISPs and edge-networks.

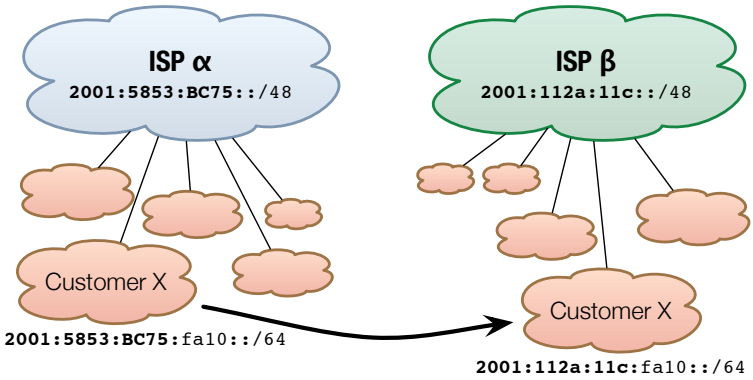
Following this scheme, each large ISP receives a /32 prefix for allocation to its clients. Based on the initial recommendations of IAB/IESG to end-sites [II01], end-sites should be assigned a /48. Based on experience, RIRs have started revisiting these recommendations to assign smaller block size to end-sites. However, a key principle is that end-sites should always be able to obtain a reasonable amount of address space for their actual and planned usage. One particular situation that must be avoided is "having an end site feel compelled to use IPv6-to-IPv6 Network Address Translation or other burdensome address conservation techniques because it could not get sufficient address space" [NHR10].

RIRs also define that a larger initial or subsequent allocation than /32 can be requested if the organization shows that it would require a larger allocation for migrating its current IPv4 services to IPv6 or for a growth plan [CSM+09, APN10]. This criterion is measured using the HD-Ratio on /56 prefixes. The HD-Ratio, the Host-Density Ratio [DH01], is a logarithmic-

[4]The *default-free zone* refers to the set of Internet backbone's routers that do not use a default route to route packets to any destination.



(a) With *Provider Independent* (PI) addressing, a customer can change its provider without changing its IP prefix.



(b) With *Provider Aggregatable* (PA) addressing, a customer moving to a new provider obtains a new IP prefix.

Figure 1.2: Impact of a provider change on addressing.

scale measurement for address space utilization and is expressed as the ratio of log(number of allocated address blocks) to log(maximum number of allocatable address blocks). APNIC defines 0.96 as the acceptable HD-ratio to justify the allocation of additional address space (/31, /30, ...) [APN10]. For RIPE, the latest document update mentions 0.94 [CSM+09], which corresponds to an utilization ratio of 36.9% for a /32 network.

1.2.2 Allocation of an IP prefix in a Network

Until now, the allocation of an IP address block within a network is mainly a manual operation. Network administrators typically prefer to keep full control on this critical step. But with IPv6, it is changing.

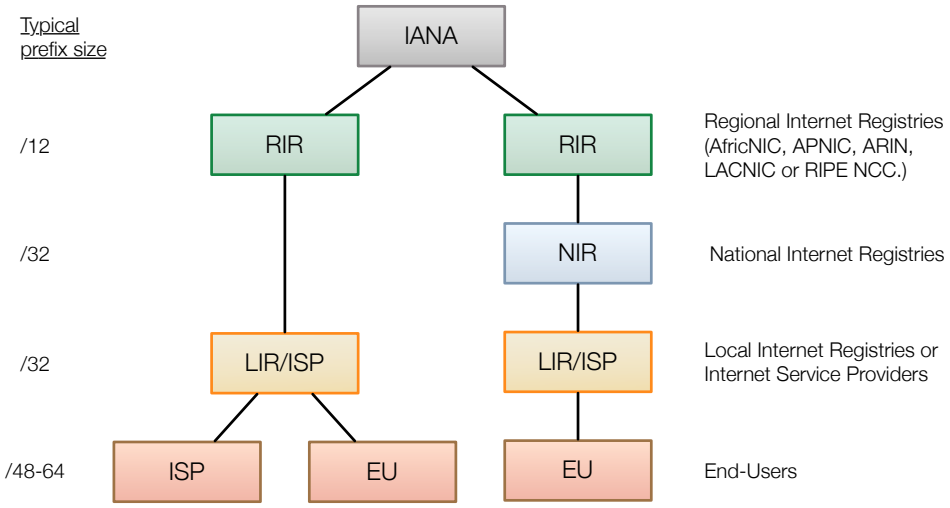


Figure 1.3: IPv6 Address allocation hierarchy and policies on sizes

Firstly, if the IP prefix of the network is likely to change, updating the configuration purely manually of the whole network can be painful. Actually, in most networks, IP addresses appear in much more places than simply in routers. The other network devices such as packet filters, DHCP servers or DNS servers typically deal with IP addresses in their configuration. Mahajan et al. [MWA02] have also shown that manual router configuration can be error-prone.

Secondly, the larger address space of IPv6 allows a much lower utilization ratio. Manual optimization of the address space is probably not required anymore in most IPv6 networks.

Unlike IPv4, it is possible in IPv6 to use more than one IP prefix at the same time in a network. There are many reasons to enable two (or more) IP prefixes in a network. Here are some of them:

- use a ULA prefix for security and reserved for internal communications;
- using two PA prefixes from two different ISPs to improve the network reliability (using SHIM6 [NB09] on end-hosts) or to perform load balancing;
- using two prefixes for a short time to replace smoothly a prefix by another one.

A last difference between IPv4 and IPv6 for the address allocation within a network is that, due to the address format (shown at Figure 1.1), the task is actually limited to allocating a *subnet id* to each subnet of the network. It is much simpler than in IPv4 where each subnet may have a different size from the other ones.

1.2.3 Host IP Address Allocation on a Local Link

New IPv6 standards have quickly defined new ways, stateless ones, for announce end-host to obtain an address on the LAN [NNSS07, TNJ07]. The egress router[5] periodically sends a *Router Advertisement* message on the LAN to announce its IP prefixes (/64 prefixes). When end-hosts receive this message, they generate a 64-bit IID to create a full IPv6 address. In order to check whether the address is already in use, the host sends a *Neighbor Solicitation* message (whose role is quite similar to ARP in IPv4) on the LAN. If it has not received any response, the address is considered unique and is used by the host. This protocol, the Neighbor Discovery Protocol, was supposed to be the only way to allocate address on a LAN using IPv6 but actually three main problems were raised. The first one was that most network administrators do not want to lose control over the addresses allocated on their LAN and prefer a stateful mechanism in which IP-MAC mapping is known and possibly fixed. The second one was that the mechanism allows to impersonate easily the egress router, and to prevents a host from obtaining an address on the LAN. Finally, ND did not permit to learn the DNS server of the network.

The first problem led to re-definition of the DHCP protocol for IPv6 [DBV+03]. DHCPv6 is stateful and works roughly in the same way as DHCP for IPv4: a server located on the LAN distributes IP addresses from its pool on request for the end-host.

The second one was solved by adding authentication and integrity checking to ND. The SEcure ND protocol (SEND) [AKZN05] uses Cryptographically Generated Addresses (CGA) [Aur05] and a signature in each critical message to achieve these goals. CGA are used to bind the IP address chosen by a host to its private key. It prevents an attacker from pretending

[5]The egress router is the first router on the outgoing path from the end-hosts to the Internet.

that each IP address proposed by a host collides with its own address. To perform authentication, SEND relies on asymmetric cryptography. Each end-host generates its own self-signed certificate and derives its (CGA) IP address from the public key using a one-way function. Then, it is able to sign ND messages sent on the LAN. *Router advertisement* messages that contain the prefix and the gateway to use must also be protected. To achieve this goal, routers also possess a certificate. But this time, this certificate must be approved by an entity trusted by the end-host. To permit deployment in a large network, a PKI and certification path can be used even if the end-host has not IP address yet.

For the third problem, ND specification mentions that DNS server information should be distributed using a DHCP server. Actually, a flag in the *Router Advertisement* message can be included to tell the end-host that other information (such as DNS server list) is available via DHCPv6. The main reason is that, for the implementation on the client side, DNS information must be treated in user space while other ND features have to be implemented in the kernel [Jeo06]. However, one may think that deploying a DHCP server only to send DNS server information is a bit overkilling. That is why a DNS configuration option has been defined [JPBM10] recently.

1.2.4 IPv6 address lifetime

To help end-hosts to deal with multiple addresses, each IPv6 address is associated with a status [TNJ07]. An address can either be tentative, preferred, deprecated, valid or invalid, as depicted on Figure 1.4. *Tentative addresses* are only used with ND (or SEND) to specify that the address has been chosen but is waiting for potential collisions. When an address has been received using DHCP and tentative time is over, address is defined valid until the end of the *valid lifetime*. This lifetime as well as the *preferred lifetime* are received within the *DHCP offer* or the *Router Advertisement*. *Valid addresses* can be used by the host and should be routed by the network. During the *preferred lifetime*, the use of the address is unrestricted both as source and destination. When the *preferred lifetime* is over but not the valid one, the address is considered as *deprecated*, which means it should not be used to initiate new connection.

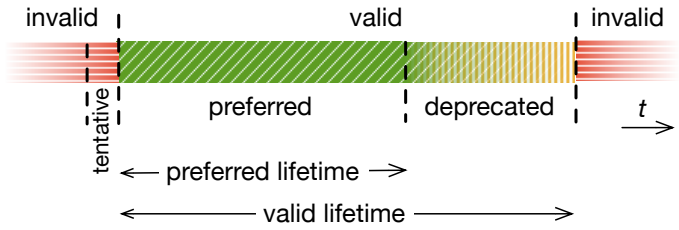


Figure 1.4: Lifetime of an IPv6 address

The status of addresses indicates to upper layers which ones can be used to initiate new connections, as defined in the address selection algorithms [Dra03]. Of course, it is also very helpful to renumber a LAN since it allows the prefix to be smoothly changed as it will be described in Chapter 3.

1.3 The Renumbering Issue

To renumber successfully, and so, to avoid provider lock-in, sites must indeed be able to announce a new prefix through their networks and to update all configuration files in which IPv6 addresses appear. Typical examples include DNS configurations, router configurations, firewall access lists, DHCP configurations, etc. For small networks with one or a very few LANs, renumbering is often not really a problem, only medium to large sites are therefore affected by this problem. The most common reasons to renumber such a network include: [CAF10, FB97]

- change of ISP or prefix change by the ISP itself,
- change of site topology (reorganization, merging, split),
- during IPv6 deployment (for the first time or when changing the access type, e.g., from tunneled to native)

Although this problem had been identified [CR96, BFLN96] in the mid 90s, most network operators think that the procedures that have to be followed today for renumbering [BLD05] require lots of manual operations and are therefore still complex, lengthy, and error-prone. Additionally, the fact that lots of networks have not started their transition to IPv6 yet can be

seen as an opportunity to set up clear renumbering procedures and tools that can be used from the IPv6 first configuration.

Most network managers agree, the renumbering of an IP network should be done without a flag day [CFV06]. A "flag day" is a procedure in which the network, or a part of it, is changed during a planned outage, or suddenly, causing an outage while the network recovers. Avoiding disruption requires the network to be modified using what in mobility might be called a "make before break" procedure: the network is enabled to use a new prefix while the old one is still operational, operation is switched to that prefix, and then the old one is taken down [BLD05]. While a flag day was needed with IPv4, the fact that end-hosts can use several addresses at the same time makes smooth renumbering feasible with IPv6.

It is worth noting that several network architecture proposals would not require edge network to renumber anymore: Identifier Locator Network Protocol (ILNP) [Atk10], Locator/ID Separation Protocol (LISP) [FFML10] and Six/One [Vog09]. However, these protocols need important architectural modifications and are far to be widely used across the Internet. Moreover, these would only eliminate the "PA addressing" reason for renumbering, not the other ones (i.e., changes in network architecture).

In this part of thesis, we propose some mechanisms and tools to ease the renumbering of an IP network. Chapter 2 describes a secure mechanism for address allocation and distribution across a network and between several networks. Chapter 3 explains how it is possible to automate the update of configuration files by using macros and simple scripts. Finally, Chapter 4 concludes this topic and discusses some possible further works.

Chapter 2

Address Allocation and Propagation

As explained in the previous chapter, address allocation and the propagation through the network is performed manually in most networks. In this chapter, we propose a new address block allocation and distribution protocol that has been designed to be both secure and efficient. We call it SMAAD, for *Secure Mechanism for Address Block Allocation and Distribution*.

SMAAD is targeted for edge networks as well as ISP networks that receive one or several prefixes from their ISPs and need to provide address blocks to some sub-networks. We model such networks as represented at Figure 2.1. In this figure, arrows represent the direction of address block allocation. SMAAD is composed of three main parts. One or several prefixes are obtained from upstream ISPs by border routers (α, β on Figure 2.1). Sub-networks needing an address block ask for it at border routers (D, E, F on Figure 2.1). The routers negotiate which parts of the obtained prefixes have to be allocated to sub-networks. These sub-networks can be a simple LAN, a self-managed sub-network of the company or customer networks which need a prefix. Actually, most edge- and ISP networks that are in charge of several sub-networks can fit in this network model.

In this chapter, we first define which other existing tools can be used to propagate prefix information through networks. Then, we describe a mechanism that would permit to fully and securely automate prefix allocation and propagation. We insist on the fact that the address allocation means,

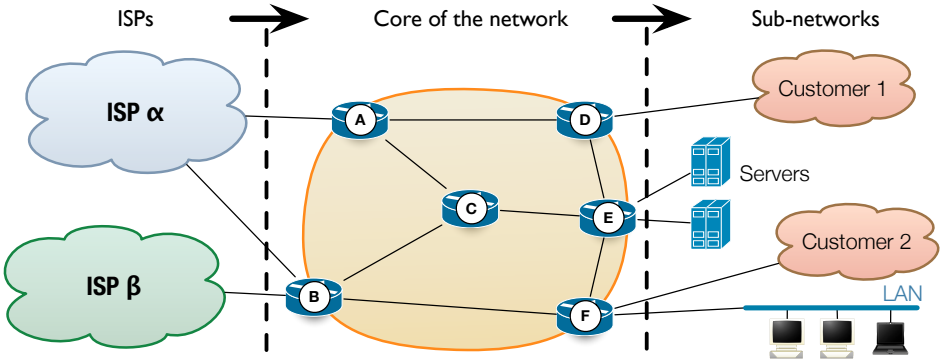


Figure 2.1: The protocol does apply on hierarchical topologies

in this context, the subdivision of a prefix received into sub-prefixes.

2.1 Existing mechanisms

Some researchers tackle with the manual configuration problem solution by providing "autoconfiguration". These solutions actually do not target the same networks or objectives than ours. For instance, none of them consider at the same time the two issues of allocating address blocks and distributing them. They also do not take into account that all the subnets have not necessary the same size.

The NAP Protocol (No Administration Protocol) by Chelius et al. is targeted at SOHO (Small Office and Home Office) network [CFT05]. They provide a protocol for allocating an address block to /64 LAN only. The NAP protocol is integrated in OSPFv3 [CFML08] using its link state advertisements and a prototype NAP router has been implemented. Unfortunately, NAP does not consider the problems faced by campus and enterprise networks such as the security issues and the need for regrouping some class of users into the same prefix. In our case, we do not think that forcing an edge network or ISP to use OSPF is feasible. Finally, NAP does not provide any security feature.

The address allocation problem has also been studied within ad-hoc networks and notably within the *autoconf* working group of the IETF [Bac08]. A detailed survey of the solutions that have been discussed within this working group may be found in [BCM10]. The solutions developed for wireless

ad-hoc networks cannot easily be reused for ISP networks because they assume that all routers will receive prefixes having the same size and do not take security into account. Moreover, these works are focused on ad-hoc networks and as they say "classical solutions do not work as-is on MANETs due to ad hoc networks' unique characteristics".

The *Prefix Delegation* [TD03] option for DHCPv6 can be used by an ISP to delegate or withdraw a prefix for a customer's router. This mechanism is intended for delegating a long-lived prefix only and the configuration of the assigned prefix must be done manually. The authentication of DHCP messages (as explained in the Section 21 of [DBV+03]) should be used to authenticate both the server and the client.

The *Router Renumbering* protocol [Cra00] is a ICMP-based protocol that could be used to propagate the current prefix or a new prefix to use in a network. The protocol is not in charge of address assignment, the announced prefix must be configured manually. The protocol does not embed any security functionality and the only way proposed to secure it is to use IPsec on each link between routers. There appears to be little experience with it [CAF10] and this protocol is rarely cited as an actual renumbering solution.

2.2 Requirements

An address block allocation protocol should achieve some important requirements. The main ones are the following.

Acceptable utilization ratio of address space. As explained in Section 1.2.1, RIRs require organization to meet a specific HD-ratio value to request a larger address block. A mechanism for prefix auto-assignment should allocate address blocks in a way such that an acceptable HD-ratio can be reached.

Security. It is easy to be convinced that an address block allocation mechanism is a key component of an IP network and could be the target of attacks. Indeed, this protocol, in the network configuration process, runs prior to the other IP protocols. Therefore, if this protocol

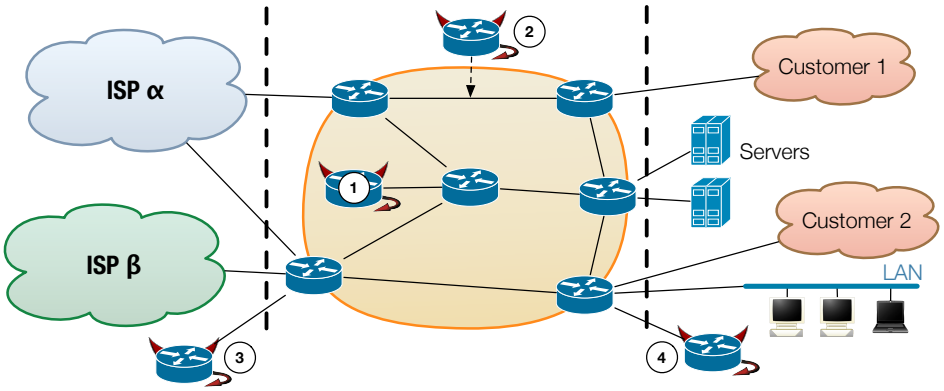


Figure 2.2: The threats to the protocol. The horned routers represent potential attackers.

were compromised, the entire network or a large part of it could be compromised.

The first risk comes from a malicious router that could be intruded into the network. In addition to the security threats on all the routing and IP protocols, this router could send several kinds of messages to confuse the address block distribution protocol. It should be possible for a router to detect whether its neighbor is a legitimate router. For instance, if router ① on Figure 2.2 is unauthorized, neighbor routers should not trust it and thus not forward its messages.

The second risk, represented by ② on Figure 2.2, is another way of obtaining similar rights but without inserting a router. It consists in intercepting the packets exchanged between two routers and performing a man-in-the-middle attack. We do not consider the confidentiality as a requirement, however message authentication and integrity must be achieved.

The last two risks are based on abusing a link between border routers and their providers or customers (e.g., as routers ③ or ④ on Figure 2.2). Invalid prefix announcements can cause a global unavailability or traffic hijacking. Abuse can also appear on a link with sub-networks, if a customer is able to request any block size without any check, it can easily exhaust the whole address space or simply obtain a prefix size larger than the one it is supposed to receive.

Compatibility with existing protocols. Since the role of the mecha-

nism is limited to address block allocation, it must be independent from the routing protocol (OSPF, IS-IS, ...). Additionally, we think that forcing a network to use a specific routing protocol to support our protocol is actually not realistic.

Roles. In an ISP, enterprise or campus network, network operators usually group the hosts that have similar roles in contiguous address blocks. For example, all loopback addresses of routers usually belong to the same prefix, all servers are grouped in a few prefixes and the hosts used by students in a campus network are grouped in contiguous prefixes as well. ISPs also group their customers in prefixes depending on their type (e.g., home users, business customers, ...). This allows defining and deploying policies on routers (e.g., QoS, traffic engineering, ...) that are more scalable and easier to maintain. Common examples are the packet filters deployed on most routers and firewalls to filter unwanted packets. Chown et al. have shown that updating packet filters is a difficult problem when renumbering a network [CFV06].

Prefix coloring. Another requirement is that the prefixes received from different providers are not necessarily equals. Due to routing policies, it can be necessary to classify the prefixes received from providers in different categories. For instance, if a National Research Network has been allocated a prefix from a backbone research network, it should only allocate address blocks from it to its customers that are research labs. Primary and secondary schools should not be allocated an address block from such a research prefix.

2.3 Description of Our Solution

Following these requirements, we have designed a new protocol that allows assigning and delivering securely new address blocks. The protocol is distributed, which means that any router that participates in the protocol is able to assign addresses. The protocol is also request-driven, i.e., it does not know in advance who will request addresses and which block size will be requested and only seek which block to assign on request.

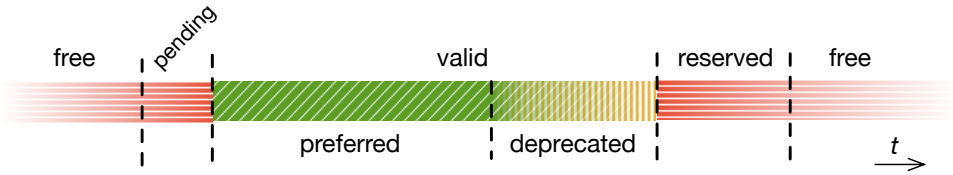


Figure 2.3: Lifetime of an IPv6 prefix in the SMAAD mechanism

2.3.1 Definitions

We allow in our protocol to request any size of address blocks. As a consequence, we need to define some new terms to define the structure of an IPv6 address as represented on Figure 1.1. The Global Routing Prefix is still the prefix received by the network from its ISP. But the network has to assign address block inside its subnet ID with respect to the block size requested. The part of the SID assigned by the network to its sub-network (the leftmost bits) will be called the "assigned SID" and noted ASID. The part of the SID that the sub-network can use inside its own network will be called the "delegated SID" and noted DSID.

Besides, we also need to introduce lifetimes for prefixes. Even if the IETF has defined some IPv6 address lifetime (as explained in Section 1.2.4), the lifetime of IP prefixes has not clearly been defined. Figure 2.3 shows the prefix statuses that are used internally for SMAAD. The new *pending* and *reserved* status work as follow: a prefix is *pending* for a short period starting when the address block has been proposed and during which collision with another prefix can occur. The *reserved* status is used for address blocks that have been assigned recently and that should not be used by someone else for a moment as some packets might still be sent from the Internet to them. Note that these two new status do not conflict with standards since they are only used locally for address blocks.

2.3.2 SMAAD in a nutshell

Disregarding the security, we can summarize the SMAAD mechanism as follows. One of the core routers receives a message from the ISP stating the prefixes that have been assigned to the network. This information is flooded to all core routers so that each one knows the information. Once a new sub-

network is connected to a core router, this sub-network's first router requests an address block of a specific size. On reception of such a request, the core router registers a new address block among the free address space. Once it has been chosen, the block ASID is broadcasted to the core of the network. If no collision is detected, the core router replies to the sub-network. This overall mechanism is similar to the NAP proposal [CFT05].

2.3.3 The secure architecture

We tackle with the security risks defined in Section 2.2 by two complementary approaches: a global top-down one in which assigned prefixes are globally authenticated and a local bottom-up one for authenticating sub-networks and core routers.

The first idea is that IPv6 prefix delegations and certificate delegations should go together. In other words, each network that possesses a prefix should obtain, from its ISP, a certificate proving that it is allowed to use and delegate it. In our mechanism, these are used to authenticate our ISP's routers and to authenticate our delegated prefixes among our sub-networks. Actually, we rely on the X.509 extensions for IP addresses [LKS04] which implies that, when one of our border routers communicates with an ISP's router, the certificate authenticates the ISP and the prefix. When a network tries to authenticate its ISP, it has typically no Internet connectivity to validate the ISP's certificate. The border router of the ISP can help its customer in obtaining evidences using a certification path, i.e., the list of certificates from the ISP to the global certification authority. This certification path can be checked by the customer in the same way it is done with other protocols already proposed at IETF [AKZN05]. Until now, this proposition only ensures top-down authentication, i.e., a network can only authenticate its ISPs, not its customers.

To allow a provider to authenticate its customers, our network has its own certification authority and each router is configured with a certificate (as suggested by Greenberg *et al.* [GHM+05]) to confirm that it does belong to the network. Moreover, an X.509 certificate signed by our local certification authority (CA) is also provided to all equipments that have specific rights in our network, i.e., all sub-networks that are allowed to request a prefix.

Actually, the certificate itself identifies its owner and we associate it with attribute certificates [FHT10] to provide authorization. These attributes are used by our subnets to specify their role, their provided prefix colors and their address block size. These certificates could be given offline to customers when the contract is signed and are used when the customer connects to one of our border routers. Attribute certificates are also used to authenticate connections among routers : a neighbor is considered as a legitimate router if and only if it can be authenticated. In practice, the CA should also be known by all nodes that authenticate other ones. The connection between two routers does not need any confidentiality but, as previously asserted, has to be protected against injections, replays and message modifications. Therefore security parameters must be used in each message sent between routers. This can be made using IPsec, SSL or by defining some new fields to our protocol messages. The revocation of the bottom-up authentication can be done easily since the core routers can receive up-to-date revocation lists or request the local certification authority.

2.3.4 Address Allocation

This section explains how address blocks are chosen within the address space for assignment. This assignment algorithm can be run by any core router that is connected to a sub-network on request from the sub-network. Its objective is to find the more suitable address block of the requested size to assign to the sub-networks.

Assigning address block in an intermediary network that has ISPs and sub-networks — as the ones we target for this protocol — can be reduced to assigning the right ASID for each sub-networks. When assigning an address block, we do not care of the prefix we receive from our ISP. Considering only ASID allows simplifying the algorithm and additionally allows making a unique ASID assignment for several prefixes, if they all have the same size. For instance, consider that the network has obtained two /32 prefixes, `2001:abcd::/32` and `2001:1234::/32`, from its ISPs. If the sub-network requests a /48 prefix and the assignment results in the "`5678`" ASID, this sub-network will obtain as prefixes `2001:abcd:5678::/48` and `2001:1234:5678::/48`.

For optimization purposes, instead of choosing an independent block for

each sub-network, each core router chooses a large address block that can contain all the sub-networks it is attached to, we will call this larger block the *router block*. This grouping allows only announcing one prefix to all the core routers, which reduces the number of messages and the table size in routers. It also allows a much better address aggregation which results into shorter forwarding table in core routers. In the topology depicted in Figure 2.1, *router F* allocates two address blocks inside its own router block : one for *Customer 2* and one for *LAN*. The trees depicted on Figure 2.4 represent, as binary trees, the typical state used for block assignment and store in core routers. In this example, the network has two /48 prefixes: `2001:4bc7:9377::/48` and `2001:3def:73cb::/48`. The left tree is the assignment tree corresponding to the *router blocks* that have been assigned. Routers A, B, C are connected to several sub-networks and have requested respectively a /50, /51 and /50 blocks. On the right part of Figure 2.4, this is the local tree corresponding to the /51 address block for sub-networks of router B, i.e., `2001:4bc7:9377:8000::/51` and `2001:3def:9377:8000::/51`. This tree, only stored in router B, details the allocation of this address block. Router B has five customers, four requesting a /54 prefix and one requesting a /53. Each customer has therefore obtained two prefixes. For instance, the second leftmost customer obtains `2001:4bc7:9377:9800::/54` and `2001:3def:73cb:9800::/53` as address blocks.

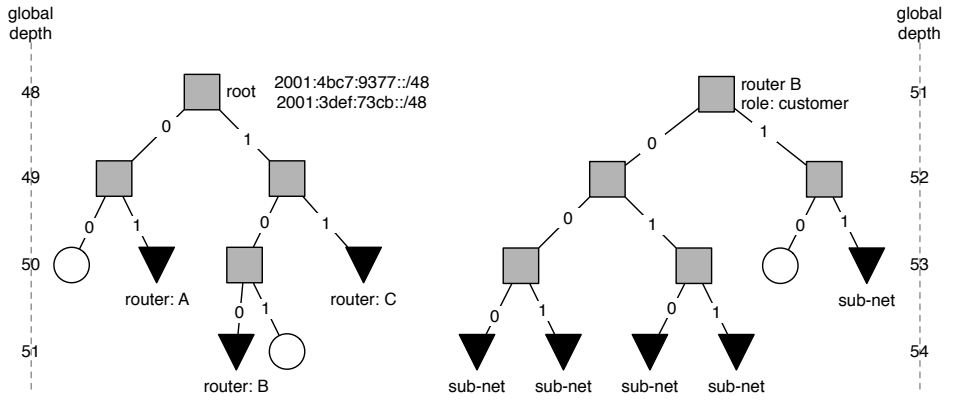


Figure 2.4: A global (left) and a local (right) tree containing some reserved address blocks. Black triangles are attributed nodes, white circles free ones and grey square partially attributed ones.

However, one question remains: *How do we choose which block to allocate among the free address space?* This is a complex question which is not cov-

ered in this thesis. Wang et al. [WGP07] proposed a growth-based address partitioning algorithm to tackle with the problem. Based on data from APNIC and CNNIC[1], they showed that their algorithm performed 90% better, in terms of fragmentation, in comparison to the current existing allocation. Recently, Murugesan *et al.* [MR10] propose a hybrid address allocation algorithm which combines several algorithms to achieve better performances. However, the lack of data about IPv6 prefix allocation makes difficult the evaluation of such methods.

2.3.5 Roles and Prefix Coloring

The roles — already addressed in Section 2.2 — are labels attached to sub-networks which allows prefix aggregation. Sub-networks related to each other, e.g., all research laboratories, can therefore be grouped in a same prefix. In our mechanism, we include this information in an attribute certificate [FHT10] attached to a sub-network's certificate. For address allocation, this information has to be taken into account. A core router dealing with sub-network with different roles must allocated two different global address blocks. The router blocks should be chosen in order to aggregate by role as much as possible.

Prefix coloring is a way to restrict some prefixes to a subset of sub-networks. This data is also determined when the sub-network first registers and can be included in an attribute certificate for each sub-network. Based on this information, core routers are able to determine which sub-network has to receive a block from an ISP prefix.

2.3.6 Address Block Distribution Protocol

The protocol for prefix delegation between border routers of different networks is outside the scope of this work. However, several already-existing protocols can be used to exchanging these prefixes. It could be designed as an extension to BGP, using DHCP with prefix delegation option [TD03] or a new protocol using the X.509 certificates being defined by the IETF *SIDR* working group. The only requirement we impose is security.

[1] China Internet Network Information Center (CNNIC) is a country registry, in charge of IP address allocation and management in China.

As previously stated, our protocol between core routers is distributed, each one is responsible for its address blocks and has to choose and advertise them. A distributed protocol avoids the single point of failure problem. Chelius *et al.* also showed that it is more efficient for address block distribution, especially if the network is multihomed [CFT05].

Each router is configured with one 64-bit router id (derived from MAC address or manually configured), its X.509 and attribute certificates and information about the subnets attached to itself. The loopback address of each router can be derived from the router id using a fixed SID for all routers.

We use a variant of flooding to distribute address block allocation information to all routers in the network. The *discover* and *hello messages* detailed below are only sent between neighbors to initiate connection. The *prefix* and *address block advertisement messages* are flooded to all routers. Except for the discover ones, messages are sent over a TCP connection, similarly to LDP [AMT07].

Discover message. This message is a router multicast[2] UDP packet periodically sent on each interface of the router. When such a message is received from an unknown neighbor, a connection is initiated with the sender.

Hello request and response messages. To initiate a connection with a neighbor, a router has to send a *hello request* message providing its ID and security parameters. If the receiver accepts the connection, it replies with a *hello response* message containing its own security parameters. Once this message has been received, the connection is considered as established. The following messages will only be sent on an established connection. A router stores a list of its direct neighbors with their security parameters associated in a neighbor table.

Prefix advertisement message. When a border router learns that a new address block has been allocated to the network, it floods immediately the information. Each prefix is associated with a preferred and a valid lifetime and so must be regularly renewed as long as it remains valid. A *prefix advertisement message* contains : the size of the prefix, the prefix itself, the deprecated and validity lifetime, the prefix color, a sequence number and

[2]Router multicast is a particular link-local IPv6 address that should be listened by routers on a LAN and that must not be forwarded [HD06].

security parameters. The sequence number is incremented each time a new message about a specific prefix is generated. Until the lifetime expires, an entry per prefix is stored in a prefix table in each router.

Address block advertisement message. When a router chooses new address blocks, it floods an *address block advertisement message*. This message contains a list of address block entries, one for each entry it chooses. When such a message is received, the proposed address blocks are checked and new address blocks are stored with the already allocated ones. An *address block advertisement message* contains the router id of its issuer and a list of address block entries. An entry contains a role, the prefix size, the ASID and its size, a timestamp, the preferred and valid lifetime and a sequence number. As for prefixes, the sequence number is incremented each time an address block entry is changed (ASID changed, preferred time redefined, ...). As the mechanism is distributed, there is a non-zero probability that two routers choose conflicting blocks nearly at the same time; we call this a collision. These collisions can be resolved by using logical clocks [Lam78][3] and randomness.

2.4 Conclusion

The SMAAD solution presented in this chapter has introduced a new paradigm for address assignment and propagation. It introduces a dynamic (as in DHCP) and a secure way to request address blocks between networks, mostly for those with a customer-provider relationship.

The security requirements are solved by using certificates based on two PKIs. The first one is rooted at a global Internet-wide certification authority. The second one is rooted in an authentication server managed by the network itself. Attribute certificates are associated to classical X.509 certificates to grant specific rights.

The address allocation and the address distribution protocol are distributed between core routers of the network. Address blocks are allocated in the address space by grouping customers by routers and applying address allocation algorithm proposed in [WGP07, MR10].

[3] Logical clocks allow to determine a partial order between events in distributed systems using counters with message passing.

In practice, some obstacles to the deployment of such a mechanism may appear. Firstly, it is only of interest for networks that need to change their prefix from time to time or ISPs that have frequent changes in customers, i.e., new customer, removal, growing, prefix length reduction ... Secondly, it proposes full automation in a domain in which everything has always been done by hand (choice of the address block, configuration of border routers). A disfunction in such a protocol might cause very serious disruption, which is very undesirable for ISPs.

SMAAD tackles with the address allocation and propagation issues but not with what is done with these new addresses within servers, firewalls and LAN routers. These devices currently contains — potentially long — configuration files that include lots of IP addresses. This problem is covered by the next chapter.

Chapter 3

Updating configuration files

We saw in the previous chapters that in addition to the assignments of new IP prefixes, another problem that is left unsolved for renumbering is the update of configurations. IP addresses often appear in these configuration files, which may cause some issue when prefixes are changed. In this chapter, we show that, independently from the SMAAD protocol, it is possible to partially automate the update of the configuration files during a renumbering process. To do so, we propose to write directly configuration filed — or to update the already written ones — to make them independent from the currently available prefixes in the site. In practice, those are written in a format that allows our script, given the current prefixes in use, to generate actual configuration files that can be used in unmodified applications. In this way, the renumbering problem is split into an addition and a removal problem and, each time a prefix change occurs, the actual configuration files are regenerated. This technique is mainly targeted to networks that aim at frequent renumbering but can be applied in a larger scope.

This chapter is organized as follows. Section 3.1 explains how a network can be prepared once for future renumbering through several case studies. Section 3.2 describes how, in a prepared network, a prefix can be added or removed. Section 3.3 describes a case study. Finally, Section 3.4 concludes this proposal.

3.1 Preparing configurations for prefix changes

An IPv6 site desiring to be prepared for an easy addition or removal of its public prefixes has to be configured in an appropriate way. This configuration is composed of two parts.

Firstly, in addition to its public IPv6 prefix or prefixes, we propose that the site also permanently uses a ULA (*Unique Local Addresses*, already explained in Section 1.1.1) prefix. The ULA prefix is used to provide stable IPv6 addresses at least to all nodes containing a configuration file that would be affected by a prefix change (routers, servers, firewalls, middleboxes, ...). Most regular IPv6 hosts either use autoconfiguration or DHCPv6 and thus do not contain configuration files that must be updated. These stable ULAs are of course only reachable inside the site but could be used for internal DNS requests, some routing protocols, internal web servers, ...

Secondly, all the configuration files are prepared for the changes. Basically, we introduce macros in the configuration files that allow generating automatically updated configuration files containing the actual IPv6 prefixes whenever a prefix is added or removed. Actually, a renumbering event contains a period during which both prefixes coexist, the new prefix being preferred to the former one for new connections. Once most connections have switched to the new prefix, the former one is removed.

This technique cannot be used in networks in which some long-lived connections must survive the renumbering. However, Quan *et al.* have shown that flows lasting 100 minutes or longer make up only 2% of traffic, and that the encountered protocols are instant messaging (MSNP, AOL), time synchronization (NTP) or multicast control (SD, PIM, SAVv1) [QH08]. Long instant messaging and time synchronization connections do not actually suffer from disconnection and should resume without pain. Multicast control protocols and BGP sessions, in which a disconnection is more troublesome, should use private addressing to avoid renumbering or could consider using protocols such as HIP [MN06] that allows changing IP addresses without disrupting the IP connections. Alternatively, migrations from one prefix to another one of the latter services could be managed manually.

A lot of commonly used services contain IP addresses in their configuration files. The following non-exhaustive list of such services is adapted

from [CFV06, CAF10].

- **DHCP server:** IP pool and DNS server(s)
- **Router Advertisement or SEND server:** the prefix advertised on the LAN
- **Packet filters and Firewalls**
- **DNS server:** Both IP-name mappings and the configuration of the server (e.g., access list)
- **Servers or network services in general:** Most servers (e.g., SMTP) allow to configure IP addresses for access control list or some other options

In the following, we concentrate on the configuration of the services that cause most problems [CFV06, MZF07] when an IPv6 network needs to be renumbered, namely DNS, firewalls, DHCP and Neighbor Discovery. For each of these services, we obtained ISP and campus network real configuration files. Similar solutions can be developed for other services. We succeeded in rewriting all these configurations using our macros and without changing their semantic. For this purpose, we developed a toolbox called *Macro-based Prefix Updater* (MPU) that can be downloaded from our website (http://inl.info.ucl.ac.be/MPU). The only requirement for our solution is that the configuration of each service can be generated from an ASCII file.

3.1.1 Configuration file dependency to IP prefixes

At first sight, using macros to rewrite configurations so that renumbering is easier may seem trivial. However adding (respectively removing) an IPv6 prefix from the configuration files used by a site is more complex than simply performing search and duplicate (respectively remove) in all configuration files. The following describes the different use cases that can occur in configuration files. For each, we discuss how it should evolve when a prefix is added.

Basic duplicate rule

Here is a resource record (i.e., a name to IP mapping) that can be encountered in a DNS zone file:

```
www    IN     AAAA    2001:1111:2222:3333:444:5:66:7777
```

In this simple example, if a new prefix is added, the record must be duplicated by replacing the current prefix of the address (e.g., the first 48 bits) by the new one. For instance:

```
www    IN     AAAA    2001:1111:2222:3333:444:5:66:7777
www    IN     AAAA    2001:aaaa:bb:cc:444:5:66:7777
```

The DNS server will then return both records if "www" is requested.

The following firewall rule, using the *Netfilter/iptables* syntax, states that TCP segments to port 80 and the `2001:0db8:100::/48` prefix must be accepted. Once more, the rule can be simply duplicated with the new prefix.

```
-A FORWARD -p tcp -m tcp --dport 80 -d 2001:0db8:100::/48 -j ACCEPT
```

Duplicate the argument of the rule

The following firewall rule is similar to the previous one but only accepts packets that are not destined to the specified prefix.[1]

```
-A FORWARD ! -d 2001:0db8:100::/48 -j ACCEPT
```

In this case, duplicating the rule is not a solution anymore. Due to negation, we need to duplicate the match. We obtain the following rule which matches when the destination differs from both the first and the second prefix.

```
-A FORWARD ! -d 2001:0db8:100::/48 ! -d 2001:abc:123::/48 -j ACCEPT
```

Another example would be an access control list (ACL) in which all the authorized prefixes are enumerated, separated by a comma.

[1] In the *Netfilter/iptables* syntax, the exclamation mark (!) indicates a *not*.

Configuration blocks

The following RAd (Router Advertisement Daemon) configuration block defines the parameters that apply to each prefix.

```
prefix 2001:0db8:0100:f101::/64 {
                    AdvOnLink on;
                    ...
};
```

If a new prefix is added, the whole block must be duplicated.

The value is not necessary the prefix value itself

Besides classical records, a DNS server defines records for reverse bindings, i.e., bindings from an IP address to the corresponding name. For these bindings, IP addresses are written as a DNS name, from right to left with dots to express the hierarchy. The following is the reverse record for address `2001:610:240:22::c100:68b`.

```
b.8.6.0.0.0.1.c.0.0.0.0.0.0.0.0.2.2.0.0.0.4.2.0.0.1.6. [...]
[...]        0.1.0.0.2.ip6.arpa. PTR        www.ipv6.ripe.net.
```

As explained earlier, IPv6 prefixes are usually associated with a preferred and valid lifetime. These lifetimes should be specified in the RAd and DHCP server configuration files. DNS records are also associated with a time-to-live (TTL). The TTL value does not have to be equal to the prefix lifetime but it may be a function of it. In particular, if the prefix is going to be removed in a few days, the TTLs of names should be lowered. Here is such record with a lifetime of 86400 seconds.

```
www    86400    IN     AAAA    2001:1111:2222:3333:444:5:66:7777
```

In both cases, adding of a prefix cannot be performed as a duplication followed by the replacement of the previous prefix by the new one. Some transformations may be needed and additional information must be retrieved to generate the new rule.

Single address options

Some configuration rules use a single address as parameter. For instance, the following line from a DNS server indicates which source address should be used to served queries. There can have one, and only one, *query-source* rule in DNS settings.

```
query-source address 2001:db8:1:a::d;
```

When a new prefix is added, this rule cannot be duplicated. However, between two prefixes, the choice is not necessarily random. For instance, a prefix that is going to be deprecated should not be preferred for use in such a rule.

Apply some rules to specific prefixes

For networks that use several IP prefixes at the same time, it can be useful to define a different policy for each prefix. For instance, a corporate network attached to one research and two commercial ISPs may use all prefixes inside its research lab while the configuration of its datacenters will only use the prefixes allocated by the commercial ISPs. Other examples are firewalls where access-lists can contain rules that are applicable to some prefixes but not all them.

3.1.2 Macro definitions

We have just seen that even if the problem may seems easy at first glance, a lot of configuration files often require some tricky operations to be updated when a new prefix is added. To cope with these issues, we propose flexible macros that are used to replace all prefixes in the configuration files.

The simplest statement in our macros is written as:

```
[[<expression1>$$<expression2>]<separator>]
```

This notation means that the string between the inner square brackets has to be repeated for each prefix after having replaced `$$` by the prefix. The separator (`<separator>`) is placed between each repetition. Typical separations

are a colon, semicolon or a new line. For example, `[[address $$:a::1],]` in a network using the prefixes `2001:db8:1::/48` and `2001:db8:8::/48` would be converted to `address 2001:db8:1:a::1,address 2001:db8:8:a::1`.

Besides, for configurations that need to apply a function on prefixes such as DNS using the reverse IP notation, a function can be defined as follows:

`[[`*<expr>*`$ `*[<function_name>:]*`$`*<expr>*`]`*<sep>*`]`

For example, `[[zone "$reverse:$.ip6.int" {...}]]` can be used to define a reverse DNS zone for each prefix of a site. Other functions are defined later for lifetimes. MPU can easily be extended with new functions.

Finally, we introduce colors to be able to discriminate the prefixes to use inside each macro. For this purpose, each prefix is associated with one or several colors. The colors to apply to a rule are defined between the two opening brackets, for instance, $[a_0, a_1, a_2, a_3; b_0, b_1, b_2[\ldots$ This statement defines two sets; set 1 and set 2, separated by a semicolon and that can be used in the expression. The first set is made of all the prefixes that have one of the colors among a_0, a_1, a_2, a_3. If the list is empty, a set covers all prefixes. For example, consider a network having both research and commercial prefixes. The research prefixes are tagged with the *rd* color while the commercial prefixes are tagged with the *comm* color. Assume that a part of a configuration file should only be applied to research prefixes. Colors can be used to restrict the duplication of this part only to research prefixes. For instance, a statement starting with "`[comm,rd;rd[`" means that two color sets are defined. The first one contains prefixes of color *rd* or *comm*. The second set only contains the prefixes having color *rd*. Given these sets, the statement can use the patterns `$1$` and `$2$` to relate respectively to prefix set 1 or 2. As a result, the statement is first duplicated for each prefix of set 1 replacing `$1$` by the corresponding prefix, then it is done for set 2. If, at one given moment, the network has one research prefix `2001:db8:3::/48` and one commercial prefix `2001:db8:7::/48`, the following firewall pseudo-rule can be written to express that all packets from commercial or research prefix to the host with suffix `::9` using research prefix have to be dropped.

```
[comm,rd;rd[-A FORWARD -s $1$::/48 -d $2$::9 -j DROP]]
```

For the sake of simplicity, no color defined between dollars is equivalent to declare set 1. In "`[[address $$:a::1],]`", it thus means that set 1 is made

of all prefixes and that this rule will be duplicated for each prefix of set 1, "`$$`" being replaced by the prefix value.

In the following, we detail how macros can be applied on DNS (Section 3.1.3), firewalls (Section 3.1.4) and some other services (Section 3.1.5).

3.1.3 The Domain Name Service (DNS)

The DNS is mainly used in networks to provide a mapping from a name to an IP address and from an address to a name. The following study is based on the grammar of ISC BIND, the most widely used Open Source DNS server[2]. IP addresses appear in three places: the server configuration, the direct mappings (name to address) and the reverse ones.

The server configuration is a set of statement blocks containing options, a sample being given in Figure 3.1. Addresses appear either in a statement definition (`server` and `zone` in Figure 3.1) or inside an option (e.g., `acl`, `query-source` or `masters`). If an address appears in a statement definition (e.g., at line 6 in Figure 3.1), the entire statement has to be duplicated for each prefix. In options, addresses are mostly used in lists, e.g., at lines 1 or 13 in Figure 3.1. In this case, each address has simply to be duplicated for each prefix. However, some BIND options require a single address as a parameter: *query-source*[*-v6*], [*alt-*]*transfer-source*[*-v6*] and *notify-source*[*-v6*]. These statements define the source addresses that must be used for specific operations. A better way to deal with it would be to rely on the host's address selection mechanism [Dra03] and thus not to use these options. If for any reason they must be used, a solution to express it with our macros is to use a color bound to one prefix, the preferred one.

The information requested and contained in DNS replies is called a *Resource Record* (RR). RRs are stored in zone files as tuples `<name> [<TTL>] <type> <value>`. Sample RRs are given in Figure 3.3. Zones are declared in the configuration file (lines 10-16 in Figure 3.1) specifying the files where RRs are stored. A zone is named by what it defines, i.e., the domain name for direct resolution and the address prefix for reverse resolution. In BIND, addresses used for a reverse zone are inverted as seen on line 15 of Figure 3.1.

[2]More precisely, the version analyzed is ISC BIND 9. It can be found on http://www.isc.org/sw/bind/.

```
1  acl "internals" { 2001:db8:1::/48; fd12:3:4::/48; }
2  options {
3          notify no;
4          query-source address 2001:db8:1:a::d;
5  };
6  server 2001:db8:1:7::a { request-ixfr no; };
7  view "local" {
8    match-clients { internals; };
9    recursion yes;
10   zone "mydomain.net" {
11     type slave;
12     file "db.net.mydomain";
13     masters { 2001:db8:1::a; };
14   };
15   zone "1.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa" { ...
16   };
17 };
```

Figure 3.1: A sample of BIND configuration file

Using our macros, the configuration of Figure 3.1 can be abstracted as shown in Figure 3.2 to become independent of the global prefixes. Line 4 of this configuration shows how an option taking only one IP address can be transformed under the assumption that the *main* color is bound to one prefix. Line 16 uses the reverse function to transform the prefix for reverse DNS zones.

```
1  acl "internals" { [[$$::/48;]] fd12:3:4::/48; }
2  options {
3          notify no;
4          [main[query-source address $$:a::d;]]
5          ...
6  };
7  [[server $$:7::a { request-ixfr no; };]]
8  view "local" {
9    match-clients { internals; };
10   recursion yes;
11   zone "mydomain.net" {
12     type slave;
13     file "db.net.mydomain";
14     masters { [main[$$::a;]] };
15   };
16 [[  zone "$reverse:$.ip6.arpa" { ...
17   }; ]]
18 };
```

Figure 3.2: The generic version of the BIND configuration file from Figure 3.1

A sample of direct (respectively reverse) RRs that can be found in zone files is shown at line 2-3 (respectively 4-5) of Figure 3.3. In direct zone files, names are mapped to IP addresses. When a new prefix is added, the RRs have therefore to be duplicated. In reverse zone files, RRs map an address suffix to a name. Those files do not need to be changed when a new prefix is added. Only their declaration statement in the server configuration (lines 15-16 in Figure 3.1) needs to be duplicated.

RRs can also contain a TTL when they are not using the default one (defined for the entire zone file), for instance 7200 and 86400 (24 hours) in Figure 3.3. This TTL gives to the DNS client the lifetime during which it can still use this record without sending a new request, typically a few days. When a new prefix is added, no special care has to be taken with this parameter. However, when a prefix has to be removed, the RRs associated with this prefix will remain in client caches for TTL time. This issue, briefly mentioned in some renumbering discussions [BLD05, CFV06], concerns only direct RRs since clients will still try to connect to hosts using addresses that do not belong to them anymore. There are two opposite solutions for this issue. First, the direct RRs could be removed DNS TTL-seconds before the actual removal of the corresponding prefix. This can only be used if there is another routable prefix in the network. Another solution is to set the TTL to zero TTL-seconds before prefix removal time and then to remove the RRs at the time of removal. This solution may lead the DNS server to receive a large number of requests during the zero-TTL period. Of course, a trade-off should be chosen between these two solutions. Such a trade-off could be to decrease the TTL in several steps, e.g., to divide the starting TTL by two TTL-seconds before prefix removal, to divide it once more by two new-TTL-seconds before prefix removal, and so on until the TTL reaches a threshold at which it is set to zero.

```
1 ns1          IN  CNAME  nameserver
2 www          IN  AAAA   fd12:3:4:5:a::9
3 www  7200    IN  AAAA   2001:db8:1:2a:a::9
4 7.1.b.a.3.a.d.0.c.7.b.e.7.9.d.1.c.0.0.0
                            IN  PTR  abcd.mydomain.net.
5 9.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.a.0.0.0
                   86400  IN  PTR  www.mydomain.net.
```

Figure 3.3: Sample BIND RRs

In practice, in Figure 3.3, since line 2 refers to a ULA prefix, the only RRs that need a change for a generic configuration is the third one which would become:

```
3 [[www  $ttl:$   IN  AAAA   $$:a::9]]
```

End-hosts may also have entries in the DNS server. If so, their entries do not necessary need to be updated using our tool when a new prefix is known. Indeed, some of them may need to use a different IP suffix for each prefix, e.g., if they are using CGA [Aur05] or privacy extensions [NDK07]. In this case, DNS records cannot be updated by simply replacing the former prefix by the new one. Dynamic updates [VTRB97] must be used by such end-hosts to update their own RRs.

If DNSSEC [AAL+05] is used in the network, any change to the zone file requires the signature of the entire zone. This signature should therefore be added to the procedures that are executed before restarting the service. For the reverse records, using DNSSEC involves, in case of renumbering, the creation of a zone that needs a new delegation from the parent server. This operation may require manual interventions.

In 2000, A6 records have been proposed [CH00] to ease renumbering. They rely on address chaining to resolve an IP address from a name. Each part of the address was placed in a separate RR, making site renumbering much easier. Unfortunately, A6 has been deprecated in favor of AAAA because of the potential overweight of the chaining [Aus02].

3.1.4 Firewalls

Other services that need to be carefully updated when a network is renumbered are the firewalls and the access-lists used on routers. Intuitively, when a prefix is removed, the rules matching the prefix (e.g., “from prefix A, drop”) must be entirely removed while in the case of rules excluding the prefix (e.g., “not from prefix A nor prefix B, drop”), only the conditions related to the prefix must be removed. When a new prefix is added, the same principles are applied, i.e., the entire rule is duplicated if the rule matches the prefix and only the condition is duplicated if the prefix is excluded.

In this section, we focus on firewall configurations that can be written as a

list of rules made of a conjunction of conditions and a target (allow or deny). The first rule matching a packet establishes the final target for the packet. In practice, all commonly used firewall implementations fit this model. We verified this on real network configurations using *Netfiler/iptables*, *Cisco IOS* access lists and JunOS firewalls. If a firewall language allows disjunctions, these can be easily transformed by splitting them into two rules as shown in Table 3.1. Like Cheswick et al. [CBR03], we use a table notation to represent firewall rules for the sake of readability.

#	action	source address	port	destination address	port
1	allow	`2001:db8:1::/48` | `2001:db8:e::/48`	*	`2001:db8:a:4::1`	`http`
2a	allow	`2001:db8:1::/48`	*	`2001:db8:a:4::1`	`http`
2b	allow	`2001:db8:e::/48`	*	`2001:db8:a:4::1`	`http`

Table 3.1: In this example, rule 1 is semantically equivalent to rules 2a and 2b. It allows `http` packets from either `2001:db8:1::/48` or `2001:db8:e::/48` prefix to a specific IP address. The vertical bar symbol (|) is used to express the *or* operator.

Formally, a filtering rule can be expressed either as the undermentioned rules 3.1 or 3.2. In these rules, C_i is a condition unrelated to IP addresses and P^A a condition matching an address or address block from prefix A. The target defines the actions to be performed when the rule is matched, in practice common targets are *allow* and *deny*.

$$\text{Either} \qquad \bigwedge^{i} C_i \wedge P^A \Rightarrow target \tag{3.1}$$

$$\text{or} \qquad \bigwedge^{i} C_i \wedge \neg P^A \Rightarrow target \tag{3.2}$$

If a new prefix, B, is added, rule 3.1 can be rewritten as follows:

$$\bigwedge^{i} C_i \wedge (P^A \vee P^B) \Rightarrow target \tag{3.3}$$

$$\text{Or} \qquad \begin{cases} \bigwedge^{i} C_i \wedge P^A \Rightarrow target \\ \bigwedge^{i} C_i \wedge P^B \Rightarrow target \end{cases} \tag{3.4}$$

This means that a rule containing a condition that matches a prefix must be entirely duplicated for each prefix. On the other hand, rule 3.2 can be

rewritten as:

$$\bigwedge^{i} C_i \wedge \neg(P^A \vee P^B) \Rightarrow target \tag{3.5}$$

Or $$\bigwedge^{i} C_i \wedge \neg P^A \wedge \neg P^B \Rightarrow target \tag{3.6}$$

Which means that a rule containing a condition excluding a prefix must have only the condition duplicated for each prefix. For instance, consider a network owning prefix `2001:db8:1::/48` that uses the filtering rules listed in Table 3.2. If this network obtains a second prefix, `2001:db8:abc::/48`, the first five rules would be updated as shown in Table 3.3. According to property 3.4, rule 1 gives two rules, 1a and 1b. On the other hand, using 3.6, rule 4 has only more conditions when a prefix is added.

The last rule (rule 6) contains several conditions related to IP addresses. When a prefix is added, such a rule can be converted in two ways, either as shown in Table 3.4 or in Table 3.5. The first one consists in duplicating the rule considering each prefix independently. It does not allow packets to be sent from one prefix to the other one. The second one considers each prefix pair for building rules. In Table 3.5, this allows packets to be sent from one prefix to another one. If the rule target were "deny", the second option should have been chosen to prevent a host from bypassing the rule by sending packets from one prefix the other one. The choice among these two depends on the policy of the network and has to be made by an administrator. The second solution should be chosen unless administrators decide to deny cross-prefix connections within the network. Both can be expressed by using our macros.

So, using the macros definitions described earlier in this section, the initial firewall of Table 3.2 could be written as Table 3.6 in order to make it prefix independent without changing its semantic. In rules 3 and 4, the ampersand between closing brackets is the separator. Rule 6 uses a semicolon between the opening brackets; this expression means that both set 1 and set 2 correspond to all prefixes. As a result, if n is the number of prefixes, this rule is first duplicated n times replacing `$1$` by each prefix and then each duplicated rule is duplicated once more n times replacing `$2$` by each prefix.

#	action	source address	port	destination address	port
1	allow	*	*	2001:db8:1:a::d	dns
2	allow	*	*	2001:db8:1:7::a	dns
3	deny	! 2001:db8:1:a::d & ! 2001:db8:1:7::a	dns	*	*
4	allow	*	*	! 2001:db8:1:5::/64	ssh
5	deny	*	*	2001:db8:1::/48	telnet
6	allow	2001:db8:1::/48	*	2001:db8:1:a::4	smtp
	[...]				

Table 3.2: A sample firewall table in a network owning the prefix 2001:db8:1::/48. In this table, the exclamation mark (!) indicates a negation, the ampersand (&) expresses an *and*, and the asterisk (*) means *any*.

#	action	source address	port	destination address	port
1a	allow	*	*	2001:db8:1:a::d	dns
1b	allow	*	*	2001:db8:abc:a::d	dns
2a	allow	*	*	2001:db8:1:7::a	dns
2b	allow	*	*	2001:db8:abc:7::a	dns
3	deny	! 2001:db8:1:a::d & ! 2001:db8:abc:a::d & ! 2001:db8:1:7::a & ! 2001:db8:abc:7::a	dns	*	*
4	allow	*	*	! 2001:db8:1:5::/64 & ! 2001:db8:abc:5::/64	ssh
5a	deny	*	*	2001:db8:1::/48	telnet
5b	deny	*	*	2001:db8:abc::/48	telnet

Table 3.3: The first five rules of Table 3.2 if a prefix (2001:db8:abc::/48) has been added to the network.

#	action	source address	port	destination address	port
6a	allow	2001:db8:1::/48	*	2001:db8:1:a::4	smtp
6b	allow	2001:db8:abc::/48	*	2001:db8:abc:a::4	smtp

Table 3.4: A first way of rewriting rule 6 of Table 3.2 when a new prefix is added. It does not allow packets to be sent from one prefix to the other one.

3.1.5 DHCP and Router Advertisements

Both DHCP and Router Advertisements (RA) are used to permit end-hosts connected on a LAN to obtain and use IP addresses. In practice, the router running DHCPv6 server (DHCPs) or Router Advertisement daemon (RAd) is often the egress router of the LAN. RAd is used for stateless IPv6 address configuration [NNSS07]. It floods on the LAN the prefixes that can be used

#	action	source address	port	destination address	port
6a'	allow	`2001:db8:1::/48`	*	`2001:db8:1:a::4`	`smtp`
6b'	allow	`2001:db8:1::/48`	*	`2001:db8:abc:a::4`	`smtp`
6c'	allow	`2001:db8:abc::/48`	*	`2001:db8:1:a::4`	`smtp`
6d'	allow	`2001:db8:abc::/48`	*	`2001:db8:abc:a::4`	`smtp`

Table 3.5: A second way of rewriting rule 6 of Table 3.2 when a new prefix is added. Rule is duplicated for each prefix pair.

#	action	source address	port	destination address	port
1	`[[`allow	*	*	`$$:a::d`	`dns]]`
2	`[[`allow	*	*	`$$:7::a`	`dns]]`
3	deny	`[[! $$:a::d ]&] & [[! $$:7::a]&]`	`dns`	*	*
4	allow	*	*	`[[! $$:5::/64]&]`	`ssh`
5	`[[`deny	*	*	`$$::/48`	`telnet]]`
6	`[;[`allow [...]	`$1$::/48`	*	`$2$:a::4`	`smtp]]`

Table 3.6: Prefix independent firewall table having the same semantic as the one of Table 3.2.

by the end-hosts as well as their *preferred* and *valid lifetimes*. On the other hand, DHCPv6 [DBV+03] is used as a server based address configuration. When it receives an address request, it replies with one or several addresses and their *preferred* and *valid lifetime* (cfr Section 1.2.4).

As for DNS RR, addresses or prefixes are therefore associated with TTLs, called lifetimes. There are two possible approaches for updating them for RAd and DHCPs. The first one is to apply the same procedure as for DNS, i.e., updating prior to renumbering the lifetime to a shorter one. Another method, that does not need early scheduling, is to reduce the prefix lifetime of end-hosts just before its expiration. For DHCPv6 its can be done by using a DHCP *reconfigure message* that triggers the hosts to re-contact the server. For RA, it is done by sending new prefix information. However the RA specification [TNJ07] defines that, in order to avoid *Denial of Service* attacks, the lifetime reduction updates of unauthenticated prefixes[3] cannot be used to reduce lifetime below two hours.

RAd and DHCPs use similar configuration files. They consist of several global parameters and some configuration blocks, one for each prefix.

[3]Authentication of RA can be done using the SEND protocol.

These blocks must be duplicated every time a prefix is added. They contain obviously the prefix value but also the different lifetimes. A sample RAd configuration and its translation to a prefix-independent one are shown on Figure 3.4 and Figure 3.5.

```
interface eth0 {
   AdvSendAdvert on;
   MaxRtrAdvInterval 60;
   prefix 2001:db8:1:e::/64 {
        AdvOnLink on;
        AdvAutonomous on;
        AdvValidLifetime 360;
        AdvPreferredLifetime 120;
   };
};
```

Figure 3.4: A sample RAd sample configuration.

```
interface eth0 {
   AdvSendAdvert on;
   MaxRtrAdvInterval 60;
   [[prefix $$:e::/64 {
        AdvOnLink on;
        AdvAutonomous on;
        AdvValidLifetime $valid:$;
        AdvPreferredLifetime $preferred:$;
   };]\n]
};
```

Figure 3.5: The configuration from Figure 3.4 rewritten using macros.

3.1.6 Other configurations

Besides these services, some other configurations need some changes. The addresses of routers and servers that neither use RA nor DHCP have to be updated. Such configuration files can be very different from one operating system to another. Fortunately, these are quite simple and similar to DHCPs/RAd configurations.

Routing protocols daemons and other strictly local protocols should use as much as possible local addresses [HH05] for their communications. However, even if global addresses are needed, we observed that configuration files of most of them are very similar to the ones we previously discussed in this

section.

3.1.7 On the cost of preparing configurations

In pure manual renumbering of configurations, each time a prefix change occurs, all the configuration files concerned have to be updated. With the solution discussed in this thesis, the renumbering has to be prepared only once, prior to any prefix change. Later, the changes can be automated and should be error free. In fact, the configuration files using macros are parsed by MPU that generates real configurations containing all the prefixes in use. The first preparation of the configuration files can be made by converting existing configurations to prefix independent ones as it has been done for DNS (from Figure 3.1 to Figure 3.2) or for firewalls (from Table 3.2 to Table 3.6). Prefix independent configurations can also be written directly from scratch when new services are configured.

It is pretty difficult to evaluate the complexity of building these macro-based configurations. When they are written directly from scratch, we can assume that the additional cost is negligible compared to the cost of writing the full configurations. The cost of modifying existing configurations seems to be fairly low. Actually, most of the process can be automated by using pattern matching scripts. To validate these claims, we obtained ISP and campus network configuration files from Japan, China and Belgium. In these files we examined during our analysis on DNS, firewalls and DHCP configurations, we observed ratios of respectively 250/253 (98.8%), 689/750 (91,7%) and 50/50 (100%) of statements that have been automatically transformed using simple scripts. The remaining ones are the more tricky one such as the Table 3.4 versus Table 3.5, or the single address options. For those, we had to take a look to interpret them in the right way.

3.2 Prefix addition or removal procedure

Section 3.1 explained how the configuration files should be prepared once, to any prefix change event. Here, we discuss how a prefix addition or removal should be performed to avoid service outage. In practice, a prefix change cannot be considered as an instantaneous event as different service updates

must be done in a specific order. For instance, a new prefix should not be assigned to end-hosts while a firewall is still blocking this prefix. As a result, a prefix addition or removal procedure comprises several steps [BLD05]. This is often error-prone and should not be done manually.

The easiest way is probably to rely on a centralized management tool that knows which services are running on which nodes and that is able to plan and monitor the renumbering procedure. In practice, this tool sends orders to network nodes according to the current state and monitors the results.

3.2.1 Related work

Detailed renumbering procedures have been discussed in several IETF documents [BLD05, CFV06] describing issues that may be raised by renumbering and existing solutions. They describe constraints, that are taken into account in this thesis, but do not propose any concrete automated configuration update mechanism.

For the distribution of prefix information within a site, distributed mechanisms could be used such as Router Renumbering [Cra00] or the ones included in full renumbering propositions described in Chapter 2. DHCPv6 using Prefix Delegation [TD03] could also be used as a centralized mechanism. However, all these solutions do not enable transmission of the information such as colors and distributed ones are probably not suitable for triggering each part of the update.

Finally, the monitoring part has been addressed by Beck et al. with the NetSV tool [BCF06]. They present an implementation of a monitoring framework and share the experience gained in its deployment. We think that based on this, similar custom tools could also be implemented without needing a complicated development process.

3.2.2 Steps of a renumbering procedure

A renumbering procedure can be separated into several phases. These are represented on Figure 3.6 in which numbers from 1 to 5 correspond to periods whereas A, B and C correspond to punctual events during the procedure.

The following list briefly summarizes the different periods and the operations to be performed during each of them [BLD05].

① The former prefix is used and the time of the next renumbering event is unknown.

② A renumbering event will soon happen. In preparation for the renumbering, times-to-live (TTL) associated with addresses of the former prefix must be reduced. The duration of this phase depends on the TTL values.

③ Configurations are updated to take into account the new prefix in addition to the former one. The new prefix is not preferred for usage for both incoming and outgoing connections. The order in which updates are performed is of concern.

④ If applicable, configurations are updated in order to prefer the new prefix to the former. Then, a transition period has to be observed so that the connections (as viewed at the transport level) using the former prefix stop and most short caches expire. This later delay is discussed in more details in Section 4.1.2.

⑤ The old prefix is removed from end-hosts, routers and all other configurations. As in phase ③, the order in which configurations are updated is important.

⑥ The new prefix is used while the former one does not appear in any configuration file.

The timings for **A** and **C** events are often under network administrator control and can be driven by economical constraints. Typically, using two routable prefixes at a same time may be more expensive than only one.

3.2.3 Removing a prefix in use

The removal of a prefix p is a 3-steps process: preparation for the removal, deprecation of p, and definitive removal. The preparation consists, as explained earlier, in starting to decrease the TTL of DNS direct RRs and the lifetime of prefixes advertised on LANs. Depreciation corresponds to setting

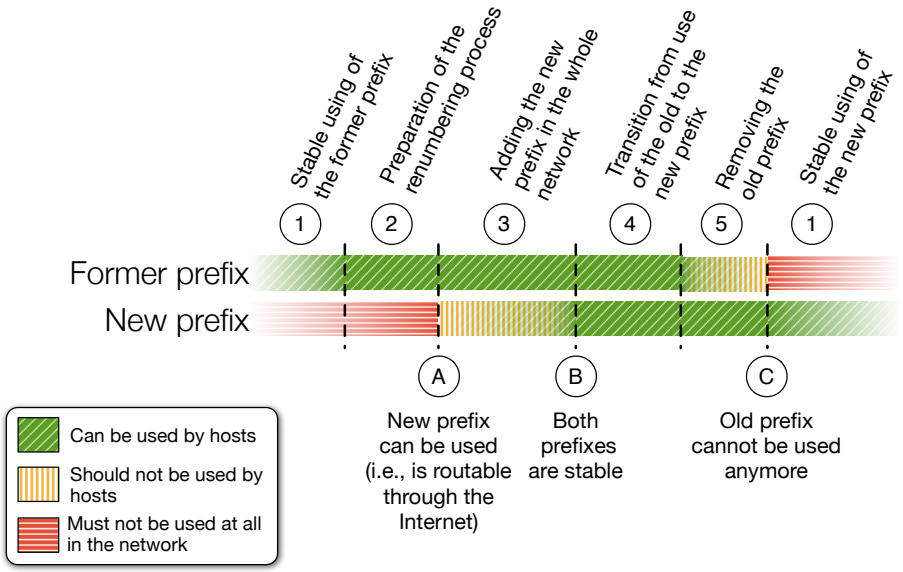


Figure 3.6: Timeline of a renumbering procedure (widths are not related the actual durations) [BLD05]

p as deprecated so that no host uses it anymore for new connections. This has to be done for RAd and DHCPs configurations as well as for static IP configuration on all network devices. DNS server configurations also have to be updated so as not to use p for requests (e.g., via the *main* color in Figure 3.2). Finally, direct RRs using p have to be removed from the DNS. These four updates have no precedence constraints among them.

The removal of all addresses related to prefix p in configuration files is the last step of the procedure. Before starting this step, it must be ensured that DNS direct RRs concerning p are not used anymore. In theory, applications should not use an address with an expired TTL anymore. In practice, some applications and some recursive DNS servers do not observe this TTL at all. As a result, some additional time should be waited before starting this last step to avoid service outage. First, both addresses on LAN (via RAd and DHCPs) and on statically assigned interfaces have to be removed. As explained in Section 3.1.5, LAN updates may need to wait for some prefix lifetime to expire on end-hosts. Once these are done, last removals can be performed, in any order, in DNS server configurations, firewalls, routing protocols and reverse zone DNS records.

3.2.4 Adding a new prefix

Adding a new prefix in a network is simpler than removing it. First of all, firewall, routing protocols and reverse zone DNS records must be updated to consider the new prefix. Once routing has converged, addresses related to this prefix can be bound to statically configured interfaces. Then, DNS server can be configured to use this new prefix, followed by LAN configuration protocols (RA and DHCP). When the prefix is used on LANs (it could take time with DHCP if reconfigure messages are not used to trigger clients to contact the DHCP server), direct DNS records must be updated to take the new prefix into account.

3.3 Case study

In order to validate our mechanism, we applied MPU on real world configurations and integrated it in realistic scenarios.

MPU is a set of scripts, written in *Python*, that can be used to apply prefix changes in a network. It is able to translate one configuration that uses the macros we defined in Section 3.1.2 to a plain configuration file. A simple mechanism for sending prefix information using SOAP in the network is also implemented.

In our case study, a central server maintains a prefix list with the parameters such as the colors and the TTL. SOAP servers are running on the hosts that run the services we want to update. They are configured with update information for their services. This update information comprises a service name (e.g., dns_direct, firewall), pre-update commands to execute (such as shutting down the server if it is needed), configuration files to update (i.e., configuration file containing macros and destination path for final configuration file) and post-update commands. When running, servers can be triggered by the central device that provides the name of the services to update and all the prefix information.

We used configurations from one ISP and several campus networks. These include DHCP servers, BIND DNS servers, Netfilter/iptables firewalls, L3-switches and routers (Cisco and Juniper, including firewalls and other IP services). Both IPv4 and IPv6 configuration were used; IPv6 be-

cause it is the target of our mechanism and IPv4 because it permits to have large real configuration not yet existing in IPv6. We succeed in rewriting all given configurations, including some complex firewall rules, using our macros and without changing their semantic. For the services we could install on our lab, we deployed them in a network using our SOAP processes for triggering the prefix changes. The services restarted correctly and their behavior was coherent and correct after each update.

3.4 Conclusion

In this section, we have proposed an approach that relies on macros to allow text-based configuration files to be automatically updated on addition or removal of an IPv6 prefix. Our macros support prefix coloring to discriminate part of the configurations based on the prefix type. We described the procedure that should be followed when a prefix is added or removed. A prototype was implemented in Python and applied to real network configurations.

The main advantage of our approach is that it can be applied quite easily on a network, without requiring large changes in configurations nor in the deployed services.

Chapter 4

Conclusion

This part of the thesis showed that renumbering is both an important and a difficult problem in deployed computer networks. Whatever the reason — provider change, network growth, topology change —, renumbering is a step through which lots of IP networks have to pass sooner or later. It has a major importance for the scalability of the future Internet and for other network operations such as network merging. Although this problem has been known for a while [CR96, BFLN96], no adequate global solution has been found.

The first step to renumber a network is to select the new IP prefix to be allocated to the network. This work is usually done "by hand" following the RIRs' recommendations. A fully automated mechanism to achieve this goal has been presented in Chapter 2. Our solution allows dynamically choosing and propagating new address blocks for new customers or ones requesting a larger address set. It can be used for new allocation as well as for IPv6 network renumbering.

The second step is to propagate the new prefix throughout the network and its sub-networks (e.g., customers' networks). Chapter 2 presented a simple secure and distributed approach to distribute address blocks in the core of the network. We introduced simple but strong security that is required by such automated mechanisms but that is missing in most other related proposals. In Chapter 3, we showed that such propagation can also be performed using scripts executed from a management host. Nevertheless this procedure may require to use monitoring tools to check that each step

has been correctly executed.

The last step is to update end-hosts and middleboxes to allow them to use the new prefix, and eventually, to remove the former one. Although a part of network services use dynamic addresses, others — even some critical ones — are configured using statically configured IP addresses. Chapter 3 introduced a solution in which actual configuration files are generated from higher level configuration files that are using macros to abstract IP prefixes. Carpenter *et al.* mentions, in RFC 5887 [CAF10], our approach as an example that should be followed to simplify renumbering.

4.1 Perspectives

In this part of the thesis, we have proposed solutions to ease the renumbering process in IP networks. Address allocation, propagation and update of configuration files have been addressed in this purpose. However, the renumbering issue can also be handled by other means. The following first briefly introduces the possible consequences of using more names from the DNS, on the renumbering issue. Then, we identify some additional problems related to renumbering that must be addressed.

4.1.1 Increase the use of DNS names

Chapter 3 showed that IP addresses appear in many places in configuration files. Most of these addresses are used to target specific hosts. One may think that these addresses could therefore be replaced by their corresponding fully-qualified domain names. It would allow getting rid of IP addresses in configuration files. In addition to the macro solution, it would let an entity change from IP address, both prefix and suffix, without the need for configuration changes.

The first issue that appears is that most rules using IP addresses do not use end-host addresses but IP prefixes. In firewalls and ACL, often a full subnet is authorized or denied; in DHCP and RAd, IP addresses appear in the form of a range or a prefix among which leases are picked. As explained in Section 3.1.3, the DNS has not implemented any prefix resolution mechanism since the deprecation of the A6 records [Aus02].

A second problem is that, as shown in Chapter 3, the behavior to adopt when there are more than one prefix is not trivial. The handling of deprecated addresses is not always simple as well. A new Resource Record cannot be added in the DNS while the firewall has not been updated to accept incoming packets for this address, which contrasts with the fact that the firewall would use the DNS to resolve names.

From network administrators' viewpoint, it has always been perceived as dangerous to have the firewalls and access lists rely on another service such as DNS [CAF10]. Firstly, a flaw in the DNS server or any attack on the protocol would compromise the whole network security. This issue can be mitigated using DNSSEC [AAL+05] which is currently being deployed across the Internet. Secondly, name resolutions may significantly reduce filtering performances. Thirdly, cyclic dependency leading to deadlocks might appear between the name resolution performed by the firewalls and the access needed by the DNS server to resolve names.

Using names in firewalls and classical server configurations would require big efforts. This would need the softwares to be changed accordingly, which is much more complicated than updating configuration files. Of course, the records of the DNS server still have to be updated each time a prefix change occurs, but Section 3.1.3 has shown that it can be partially automated.

Using names from DNS to help renumbering seems both easy and feasible in some contexts. However, it may require changing applications to be applied in depth, which is much more difficult in practice. Even if using names should be encouraged, e.g., in conjunction with our macro-based solution, it would require more in-depth analyses to determine the cost of making critical services, such as firewalls, using only names.

4.1.2 Additional barriers to renumbering

Until now, the main reason we have given to explain why renumbering is avoided is the cost of doing it by hand. However, there are some additional reasons that might refrain from renumbering IP networks. The following describes three of them.

Long-lived connections

As already stated, an overlap between the former and the new prefix allows smoothing the renumbering procedure. A connection started prior to the introduction of the new prefix will not be interrupted if it finishes before the removal of the former prefix. Once the new prefix has been introduced as preferred, all new connections should use it. Only connections (i.e., flows) longer than these overlap period will experience problem. Very few connections live more than a few minutes and even less more than a few hours [QH08].

Transport-layer connections (TCP and UDP) are bound to an IP address pair for their entire lifetime. Since a connection was not supposed to change its endpoints, this did not raise any problem earlier. But, due to the fact that the IP address of a host is at the same time its identifier (for the transport-layer connection) and its locator (for the network-layer routing), the removal of a prefix will end all connections related to this prefix. As mentioned in Section 3.1, a simple solution to avoid breaking connections is to use both the new and the former prefix for a period during which new connections are initiated on the new prefix while the former prefix is still used for previously-established connections. However, it does not eliminate the connection cut for the flows longer than the overlap period.

Several solutions have been proposed for years to solve this issue. Shim6 [NB09] and MPTCP [FRH10] allow changing one's IP address at the network layer during a connection without changing the IP address seen at the upper layer. Unlike some people think, these protocols cannot really be used for renumbering. The reason is that even if the address at the lower layers has changed, the upper layer still uses the former address. If this IP address becomes invalid for a host, it may be reassigned to another one. This would result in several hosts using the same address in the application layer, which may raise operating and security issue if these hosts are communicating with the same host. This problem is addressed in Section *1.5. Renumbering Implications* of [NB09].

HIP [MN06] is similar but uses non-routable IP addresses at the application layer which allows it to be used during renumbering. However, HIP needs a complex infrastructure for deployment and is not supported by lots

of networks and hosts. Recently, Ubillos *et al.* proposed using name based sockets [UM10] to get rid of IP addresses in upper layer. Unfortunately, this would solve the SHIM6 and MPTCP issue for renumbering only if all applications were modified to use names instead of addresses. The Stream Control Transmission Protocol (SCTP) [Ste07] would also allow renumbering using a Dynamic Address Reconfiguration option [SXT+07] but it replaces the actual transport-layer protocols and thus also requires the modification of all applications.

It should also be reminded that we recommend using ULA for connections between two hosts that are located in the same network. ULAs are not affected by the global prefix renumbering and so, can be used for internal long-lived connections.

Reference to IP addresses belonging to one network into foreign networks

For most large networks, IP addresses do not only appear in their own configuration files but also in some configurations hosted in other networks managed by independent authorities. Most of the time, these entries have been manually entered to grant permission to a network.

For instance, let us consider an educational network that has an agreement with the neighboring university to share storage for document deposit. The network that hosts the service will have to explicitly allow the IP prefix of the other organization in the storage server and eventually in the firewall. Another example is the case of digital libraries. For the sake of usability, access to digital libraries for organization is usually done using a white access list based on IP addresses.

All these configurations, within networks managed by a third party, are much more difficult to update for an administrator than his own network. These updates are typically done manually, so using DNS would be of great help.

TTL is not always enforced

It has been shown that the time-to-live attached to DNS entries are not perfectly enforced [BLD05, CAF10, PAS+04, Nor06, Mic09]. The typical way a name resolution is requested is the following: an application on the user's computer requests that a service of the operating system resolves the name. This service sends a request to a local resolver which resolves the full name from the DNS root servers up to the requested host's DNS server. The problem is that the name resolution may be cached at each step, not always with respect to the TTL.

A local DNS resolver typically performs the full DNS resolution only for end-hosts in its own network. These resolvers often cache their DNS responses in order to serve them more quickly for the subsequent requests. However storing the address resolution and the TTL is not enough. Since the TTL is a relative value (e.g., 86400 sec), if the initial request has been done at t_1 and the TTL value is T, when a client requests the same name at t_2, the resolver must respond with a TTL equals to $T - t_2 + t_1$. Pang *et al.* have shown that about 14% of local resolvers do not adhere to DNS TTLs and that those violating TTLs do so by a large amount, in excess of two hours [PAS+04].

The Operating Systems also caches name requested by applications to avoid sending repetitively the same DNS requests to the local resolver. Some hosts have experienced issues in the past, due to bad OS implementation of the DNS cache.

Finally, when an application requests the IP address corresponding to a name[1], it only receives the address, not the TTL. The application is actually supposed to request the resolution each time a new socket is created but in practice, the IP address is often cached and used during the entire application lifetime. For example, Pang *et al.* showed that 47% of Web event clients do not observe the DNS TTLs [PAS+04, Mic09]. Once more, name-based sockets would help, but this solution does not solve the problem for existing applications.

These three places where the DNS TTLs may be misused lead to an

[1] E.g. via the `gethostbyname` function, both on the Unix and the Microsoft Windows Operating Systems

incorrect TTL usage in the end-host applications. For renumbering, we use the TTL to switch smoothly from one prefix to another. If the TTL is not enforced during renumbering, a service outage can occur. Moreover, the network administrator does not even have the control on these caches since the bad implementations may be located in his website's visitors, i.e., his potential customers.

Part II

SWISH: Secure WiFi Sharing

Chapter 5

Introduction

WiFi is today the most successful wireless network technology. The number of WiFi enabled devices continues to grow. While a few years ago WiFi was only supported on laptops, there are now WiFi cards on notebooks, cell phones, smart phones, e-readers, portable game consoles, etc. Recent surveys [Cis10] indicate that more than 50% of Internet users in Europe have a wireless network at home (up to 75% in countries such as Spain) and most companies have deployed WiFi networks on their premisses.

In parallel to this technical evolution, Web 2.0 services combined with the cloud-based services encourage users to remain always connected [Dev09]. Given the ubiquity of WiFi networks, Internet access could be provided everywhere if users were able to easily and safely share their access point. Some restaurants, bars and hotels provide free WiFi Internet access to attract users [Sha10], and in dense public places such as train stations or airports, commercial WiFi hotspot providers offer paid Internet access services. Unfortunately, openly sharing Internet access through a WiFi access point raises legal [Sí09, Hal05] and security [Hin05] issues. Indeed, studies [Dev09, Var08] reveal that many of these users would likely agree to share their WiFi Internet connectivity on the condition that access by foreign users does not cause security and liability problems.

In this part of the thesis, we propose, implement and evaluate SWISH – a novel solution that enables safe and secure WiFi sharing. SWISH relies on cryptographic mechanisms for authentication between the three entities involved in the sharing process, the visiting user, his home network and

the visited WiFi network. SWISH focuses not only on protecting the visiting user and home network against a malicious WiFi network, as could be achieved by using a VPN, but also on protecting the WiFi network against abuses by malevolent users. Furthermore, SWISH supports accounting for commercial usage and is able to preserve the anonymity of the visiting users while allowing potential criminal activities to be tracked. SWISH has been successfully implemented on stand-alone Linux-based smartphones and access points, deployed and tested in a production campus WiFi network.

In order to better understand the challenges of a secure and efficient WiFi sharing solution, this chapter first discusses potential security threats, accounting risks aspects, and operational problems including usability aspects. Then, it describes some popular approaches used in current sharing solutions and reflects their strengths and weaknesses in the light of the discussed challenges. Finally, it presents the road map for the next chapters related to the SWISH solution.

The results from this part of the thesis are part of a joint work between our research group, and Mark Manulis, François Koeune and Julien Cathalo from the *UCL Crypto Group*.

5.1 Challenges of WiFi Sharing

We have classified the main challenges of WiFi sharing into three categories: the security threats, the accounting risks, and the administrative and usability problems.

5.1.1 Security Threats

Infrastructure Attacks. Mobile users connecting to foreign networks are not necessarily trusted by the latter. Therefore, granting network access to such users is risky as this would make the infrastructure components potentially vulnerable to attacks mounted from inside the network. The reason is that after obtaining an IP address from the visited network, it might be possible for a mobile device to bypass access lists typically based on IP prefixes. Additionally, a visitor might avoid firewalls located at the border of the visited network.

Resource Exhaustion. Even if roaming mobile users receive only restricted access to visited networks, the ability to access the Internet through roaming still opens doors for denial-of-service and resource exhaustion attacks against the latter. This is possible since users share the bandwidth and consume the — possibly limited — traffic volume. This may have negative performance impact on other (possibly native to the visited network) users of the system. For example, in user-provided networking, a user may have a limit, imposed by his ISP, on the volume of data he can download and upload per month. In this case, sharing a connection with other users without having control of the consumed volume becomes risky. Hence, it is desirable that sharing solutions provide network owners with the ability to monitor and control the bandwidth as well as the traffic volume consumed by their visitors.

Blacklisting. The purpose of WiFi sharing is to grant mobile users access to the Internet using the infrastructure of the visited network. The mobile device of the foreign user typically obtains an IP address that is allocated to the visited network, or uses such IP address upon accessing sites on the Internet indirectly (in cases where the visited network uses network address translation techniques [SE01]). In both cases malicious activities of mobile users on the Internet may result in an IP trace-back leading to the visited network. For instance, if the user — intentionally or not — sends spam, the visited network's prefix may be added in spam databases or other blacklists.

Fraudulent Access Points. One of the challenges in wireless roaming is the protection of mobile users from accessing the visited network via an untrusted access point. If the mobile user is not able to authenticate the access point of the visited network then the openness of the wireless channel may result in the connection to some fraudulent access points set up by an attacker. In this way, the attacker can connect to the Internet by acting as a man-in-the-middle between the mobile user and the real visited network. Consequently, he can mount attacks on the mobile user and his communication [Swi06, Bid05, Mes10].

User Profiling and Traceability. Most wireless roaming solutions require the mobile user to supply his name, user id or credit card number upon each new connection to the visited network. This information is sometimes necessary for commercial reasons but also enables the administrator of the visited network to identify the user in case this user misbehaves on

the Internet. Thus, identifying mobile users can be seen as one of the goals of the visited network. However, such a goal may contradict the privacy demands of users to remain anonymous towards visited networks. This may be helpful for example to prevent profiling of users or location tracking, e.g., if the user connects to hotspots that are operated by the same network provider in different geographic locations. Hence, the challenge is to satisfy the privacy demands of mobile users against visited networks while protecting visited networks from problems that may arise due to misbehaving anonymous users.

5.1.2 Accounting Risks

WiFi sharing solutions deployed in commercial applications should support accounting and billing. In many available systems, measurements that are relevant to the billing are performed solely by the visited network. Depending on the billing model this imposes some risks on both mobile users and visited networks. We distinguish between pre-paid and post-paid solutions. If a mobile user pays for the roaming service in advance then a malicious visited network may refuse to provide a complete service. This is, however, a minor issue since such a misbehavior would damage the credibility of networks and result in a loss of customers. The actual problem in pre-paid solutions is their usability since users have to purchase the service in advance. In general, post-paid solutions where users receive (e.g., monthly) bills from their home network provider for all the services including roaming would be more convenient to the users. However, in this case we identify the following two risks.

Risk of Overcharge. Since visited networks typically perform accounting individually there is a risk for mobile users to be overcharged. This risk is higher compared to the unfulfilled service in pre-paid solutions since the overcharge may remain unnoticed by the user, e.g., if much time has elapsed since the roaming session.

Risk of Repudiation. Another risk in post-paid solutions for the visited network is that users may contest the bills, even if these are correct. The reason is that current solutions do not provide non-repudiable confirmation receipts to visited networks for the amount of roaming service consumed by

foreign mobile users.

Note that the above two problems result mainly from the lack of trust between foreign mobile users and visited networks that provide the roaming service.

5.1.3 Administrative Challenges and Usability Problems

In addition to the security threats and accounting risks, further problems exist at the operational level, which may also impact the usability of the roaming solution.

Application Confinement. Typically, different networks are administered on an individual basis. This means that security policies of a visited network might be stricter than those of the mobile user's home network. This may result in the application confinement, which can be seen as an usability problem with regard to the mobile user. For example, mobile users may not have full access to the applications (such as VoIP or SSH) they are used to in their native environment.

Access to Subscribed Services. On the Internet, many service providers offer access to their services, e.g., access to digital libraries, based on the source IP address. In fact, they verify that the IP address of the client comes from the domain of the subscriber network. The allocation of IP addresses to foreign mobile users by the visited network would thus provide these users with unrestricted access to such services too. This might be undesirable if the access should be preserved for users who are native to the visited network. In the same way, a roaming mobile user will not have access to the services he is used to when he is at home.

Legal Risks and Tarnished Reputation. The potential IP trace-back to the visited network in case of misbehavior of foreign mobile users mentioned in blacklisting discussion may have legal consequences [Sí09, Hal05] for the owners of the visited network. Even if some authentication mechanisms for foreign mobile users are deployed (which is a challenge by itself) it would still take time for the network provider to react to the accusations by identifying mobile users. Also the inclusion of an IP address from the visited network into blacklists may initially result in the tarnished reputation of the network

provider. Besides, it may take time and resources to remove IP addresses from blacklists.

Impact of Home Network's Policies. Security policies of home networks may impose operational challenges and usability problems for mobile users. A prominent example is when a company defines strong security policies for the mobile devices of its employees that preclude their access to untrusted visited networks. In this case, the employee may experience problems in finding an appropriate trusted network while being outside of the company.

5.2 Popular Solutions and Their Limits

There are many ways to share an Internet access with foreign users using WiFi, some are designed for companies, others for home users. However, as often in computer science, the most usable solutions are seldom the most secure ones. The following presents several popular solutions.

5.2.1 Temporary Credentials

A basic solution often used in corporate networks to handle access of foreign users to the Internet is to use temporary credentials. A specific SSID (i.e., Wireless LAN name) is usually broadcasted to simplify the isolation of mobile users. This solution suffers from high administrative costs for the management of temporary accounts and monitoring of possible misbehavior.

With this technique, the visitors are often not aware of the code of conduct of computer resources applied in the visited network. The identity of each user should also be verified and bound to the temporary user id he has received.

Moreover the visitors have typically to enter these temporary credentials on a HTTP(s) portal (e.g., *Chillipot* [Chi]) of which authenticity cannot be verified by the visitor.

5.2.2 Hotspots

Mobile users frequently access the Internet by using hotspots maintained by commercial network providers.

These solutions are often web-based such that the mobile user provides his own credentials as input to the browser form. The credentials are usually forwarded to the authentication server (e.g., RADIUS). This approach was recommended by the Wi-Fi alliance as a best practice solution [ABS03] in 2003.

It does not provide authentication of hotspots (access points) towards users. Thus, using hotspots may lead to attacks by fraudulent access points. Nowadays, it has become easy to set up such a rogue access point using toolboxes [Gra07, Wex10, Ile05].

Most hotspot solutions include accounting tools. However, the duration or data count is only performed into the visited network which cannot bring proof for the claimed value.

5.2.3 User Communities

A typical way for home users to both share their home Internet connection and take benefit from a wider access to the Internet is to become part of some user-provided networking community.

Wifi.com is a *software-based* Internet sharing solution installed on users' computers. The software handles the WiFi key sharing between the users of the community. When starting using the service, the user's WiFi pre-shared key must be shared with the community. Then, when this user approaches a WiFi AP belonging to another user of the community, the *Wifi.com* software detects it, retrieves the pre-shared key and uses it to connect to the WiFi network. In order to be able to retrieve the key offline, all keys have to be stored on each user's computer, which is far from secure.

FON is a *hardware-based* sharing solution in which users plug at home a light access point that broadcasts a closed SSID for private use and an open SSID for visitors. FON claims nearly 2 million spots worldwide. The open FON SSID hosts a web portal [Chi] on which any FON user is able to connect. People not belonging to the community are also able to connect on the access point either by paying online, or by watching a short advertisement video granting free 15 minutes of Internet access. This later access method may raise identification issues if the user creates a fake account. In non-free scenarios, the connection duration measurement is only performed

by FON which charges the visitor and redistributes a part of the revenue to the owner of the shared access point. However, the visitor does not have control on this measurement and an access point owner might drastically reduce visitor's bandwidth to increase the duration and so his revenue. Even if FON mostly target end-users, several ISPs provide their customers with *FON*-capable set-top boxes [FON].

In both solutions, sharing of Internet access is done on a cooperative basis, i.e., users wishing to use connections provided by other users must be willing to share their own connection with the community. Such Internet sharing imposes many of the above described security threats on the members of the community, both sharers and mobile users. Moreover, using *Wifi.com*, visitors are connected on the same LAN as the network owner. It means that the visitor may have access to the shared printers, files and devices connected on the LAN.

5.2.4 Eduroam

A more secure way for realizing Internet sharing between corporate partners is to allow other networks to question each other's authentication server to verify collaborators' credentials. With this mechanism, foreign users can log in on the access point of a partner network and authenticate themselves using their credentials known within their home network. These credentials typically specify the user's home network, e.g., `johndoe@abc.com`. In fact, the authentication server of the visited network delegates the authentication process to the user's home network.

This solution is implemented using EAP-TTLS [FBW08] and RADIUS servers [RWRS00] between educational institutions participating to *Eduroam* [Edu]. EAP-TTLS is the authentication protocol that is used between the mobile device and the AP. The EAP frames are encapsulated in RADIUS packets and sent to the visited network authentication server which relays them to the user's home network. The architecture of *Eduroam* is depicted in Figure 5.1. On this picture, John Doe uses `johndoe@institutionB` as a login to connect while located inside *Institution A*'s buildings. Since the user database of institution A does not have entries for this user, the RADIUS server acts as a proxy and relays the requests and responses to the

upper *Eduroam* RADIUS server, which is often a national RADIUS server. If it does not match, the request is relayed to the top level proxy, making the authentication delay longer. The request finally reaches the institution B's RADIUS server which knows John Doe and can complete the TTLS protocol with the user. We measured, in UCLouvain, that *Eduroam* needs to exchange at least 12 RADIUS packets (6 for each direction) between the visited network and the user's home network to complete the authentication.

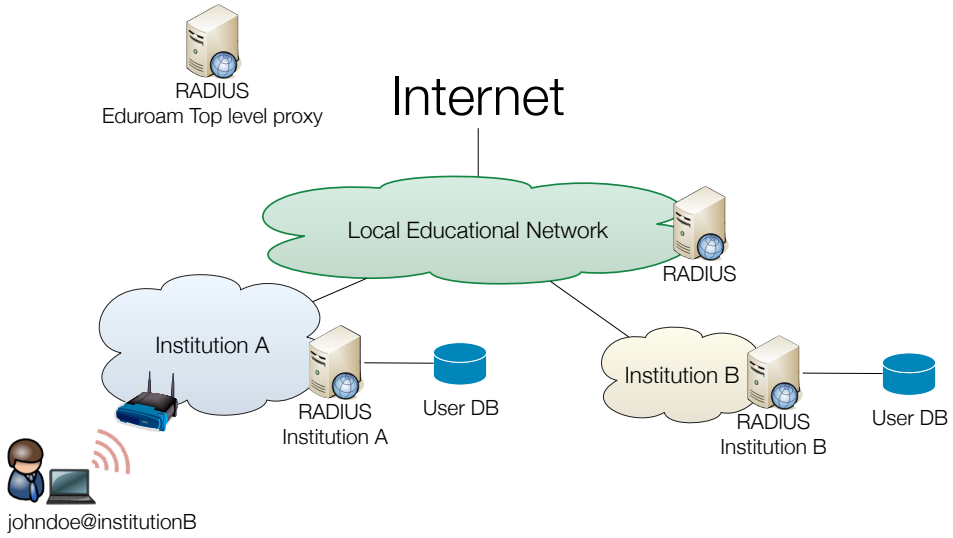


Figure 5.1: Architecture of *Eduroam*. In this example, John Doe is able to connect to the Internet through *Institution A* with his login from *Institution B*. The Eduroam authentication uses a hierarchy of RADIUS servers.

The user can authenticate the authentication server he is exchanging packets with. This authentication is included in the TTLS protocol and is performed using certificates.

Eduroam permits students and researchers to connect to the Internet for free when visiting other institutions. The operational problem with this mechanism is that it assumes strong trust between institutions, which does not necessarily fit the requirements of commercial companies. For example, an institution which is part of *Eduroam* could easily steal user credentials and may also suffer from misbehavior of visiting users.

5.2.5 Virtual Private Networks (VPN)

Many companies improve the remote connectivity of their mobile employees (e.g., road warriors, home workers) through the deployment of mandatory VPN solutions. This also allows secure remote access to the Internet via the corporate network. A VPN connection is initiated by the mobile user who creates an authenticated and encrypted tunnel to his home network. Once the tunnel is established, all data sent by the user to the Internet is transmitted through this secure tunnel to the home network, which then forwards the packets over the Internet. However, in order to initiate a remote VPN connection, a mobile user must be already connected to a visited network, have obtained a public IP and be able to access the Internet. Therefore, VPN connections are not sufficient to protect the security of visited networks against infrastructure and denial-of-service attacks or blacklisting. Additionally, since such connections are initiated by users, they cannot actually be used by the visited network to improve its security.

5.3 Summary

For each of the aforementioned popular solutions, we analyze in Table 5.1 the potential risks in the light of the threats and problems discussed in Sections 5.1.1, 5.1.2 and 5.1.3. Although our analysis may seem subjective and one may think of additional protection mechanisms that could be used to thwart one or another risk, we believe that our view reflects the general picture of how these approaches are deployed today.

As can be seen, none of the methods seems to be sufficient to simultaneously protect security and privacy of networks and mobile users, reduce accounting risks, and enable better operability and usability. The deployment of current solutions is especially risky for (non-commercial) visited networks with regard to the missing protection against infrastructure attacks, DoS attacks, and misbehaving users and for mobile users with regard to the trustworthiness of the visited networks and attacks via fraudulent access points.

Therefore, a presumably better solution would be to combine some strengths of existing WiFi sharing techniques such as those based on secure tun-

neling with extended protection mechanisms for visited networks and mobile users. This is precisely the motivation for our SWISH framework, which provides security and ease of use though a unique three-party authentication protocol and data tunneling.

The remainder of the thesis is organized as follows. Chapter 6 describes the entire SWISH solution, i.e., the mechanism, the authentication and key exchange protocol, and two extensions, an accounting mechanism and a protocol providing anonymity and untraceability to visiting users. We then discuss in Chapter 7 the implementation of SWISH, its deployment in corporate and home networks and a performance evaluation in a test bed. In the same section, we also compare SWISH with related work.

	Temporary credentials	Hotspots	FON	*Eduroam*	VPN
Infrastructure Attacks	high	none	low	high	high [a]
Resource Exhaustion	high	low [b]	low [c]	high	high
Blacklisting	high	low [d]	high	high	high [a]
Fraudulent Access Points	yes [e]	yes [e]	yes	no [f]	no
User Profiling and Traceability	none [g]	high	low	high	N/A [h]
Risk of Overcharge	N/A	yes	yes	N/A	N/A
Risk of Repudiation	N/A	yes	yes	N/A	N/A
Application Confinement	yes	yes	yes	yes	no
Access to Subscribed Services	yes	no	no	yes	N/A
Legal Risks and Tarnished Reputation	high	medium	high	high	N/A
Impact of Home Network's Policies	yes	yes	yes	no [f]	no

[a] The mobile still needs an IP address from the visited network.
[b] It is easy, for companies whose main business is Internet sharing, to prevent this using traffic shaping.
[c] FON allows to set a bandwidth limit for visitors.
[d] Sharing being the only business of hotspots companies, this threat does not have serious impact on the side services.
[e] In most cases, it is based on Web authentication in which the user cannot authenticate the visited network.
[f] EAP-TTLS uses a certificate for the visited network. However, it has to be verified by the mobile user.
[g] We consider temporary credentials distributed sequentially for a one-shot event.
[h] This depends on which connection mechanism it is based on.

Table 5.1: Summary of risks and threats in popular WiFi sharing solutions

Chapter 6

SWISH

In SWISH, we avoid many of the aforementioned security threats and usability problems by extending the roles of the networks participating in the roaming process through a novel approach. At a high level, WiFi sharing in SWISH is realized using a secure tunnel between the home and visited networks. All the packets from the mobile are then forwarded through this tunnel. Subsequently, the mobile device accesses to the Internet as if it were located inside its home network, i.e., using an IP address allocated by the latter, thus thwarting blacklisting risks for the visited network. Internet access is therefore provided without endangering the visited network.

SWISH was designed with specific attention to the usually limited computational and storage resources of mobile devices. In particular, one of the design goals behind SWISH protocols was to reduce the complexity of cryptographic operations performed by mobile devices. In this chapter we describe the overall architecture of SWISH as well as its authentication and key exchange protocol and the mechanisms used for authentication and key establishment, accounting, and anonymity. We then end with an overview of the related work.

This chapter is organized as follows. Section 6.1 defines requirements to the mechanism design. Then, Section 6.2 presents the SWISH architecture. Section 6.3 details the three-party authentication protocol, RAKE. Sections 6.4, 6.5 and 6.6 describes three extensions to the SWISH mechanism, respectively an accounting protocol, a traceability avoidance mechanism and denial-of-service protection.

6.1 Requirements

The architecture of SWISH is comprised of three main logical entities: the mobile user $\mathcal{M}$, his home network $\mathcal{H}$, and the visited network $\mathcal{V}$. $\mathcal{M}$ is located in the $\mathcal{V}$ premises and wants to access the Internet using $\mathcal{V}$'s network equipments. To tackle the aforementioned technical and management issues, $\mathcal{M}$ only obtains an IP address from his home network and is restricted by $\mathcal{V}$ to accessing the Internet via $\mathcal{H}$. A tunnel between both networks is created to redirect the packets from and to $\mathcal{M}$.

The trust assumptions amongst these entities are the following: there is a fully trusted relationship between $\mathcal{M}$ and $\mathcal{H}$. In practice, they share a common high-entropy secret psk_{MH} as a product of the registration phase. The trust level between $\mathcal{H}$ and $\mathcal{V}$ can vary depending on the roaming agreement established between both networks, either directly or through a trusted third party. In both cases their authentication relies on public key certificates. Additionally, $\mathcal{H}$ acts as an authorizer of the connection between $\mathcal{M}$ and $\mathcal{V}$. The reason is that $\mathcal{M}$ and $\mathcal{V}$ may not be aware of each other's identity prior to the roaming session so that their trust has to be established transitively via $\mathcal{H}$.

The design of SWISH is mainly driven by security requirements. These are presented through the following security goals.

Security Goals

The security goals will be addressed by considering the different kind of attacks that could be launched on such a mechanism. In the following, we consider that the attacker is located either on $\mathcal{M}$-$\mathcal{V}$ or on $\mathcal{V}$-$\mathcal{H}$ channel as a man-in-the-middle, on- or off-path.

As the mechanism aims at providing an Internet access with an IP address from $\mathcal{H}$, we want to prevent any user without any roaming agreement with $\mathcal{H}$ from obtaining access to the Internet via $\mathcal{H}$. This user may be an attacker or even be a $\mathcal{V}$ that would pretend to host a mobile user to gain the access. Similarly we want to prevent a user from accessing the Internet through a $\mathcal{H}$ which is not a partner of the visited network. Otherwise, that would allow an unauthorized user that has an associate on the Internet to

access the Internet through $\mathcal{V}$ and the associate.

In SWISH, we want to protect the data traffic from alteration and injection, both on $\mathcal{M}$-$\mathcal{V}$ and $\mathcal{V}$-$\mathcal{H}$ channels. Additionally, due to the ease of performing the attack, we require the $\mathcal{M}$-$\mathcal{V}$ channel, i.e., WiFi, to be protected against sniffing. We do not require the $\mathcal{V}$-$\mathcal{H}$ path, typically across the Internet, to be protected against sniffing but this protection can be enabled on $\mathcal{H}$'s or $\mathcal{M}$'s request.

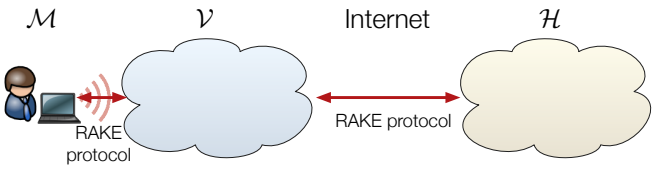
An attacker may try to pretend to be a home network to have the entire data traffic from $\mathcal{M}$ forwarded through him. In the same way, an attacker may set up a rogue access point and act as a visited network to capture packets. Obviously an attacker may also play both roles. All these attacks should be prevented.

In addition to these main goals, we introduce three side security goals. First, we want to be able to measure the roaming usage of $\mathcal{M}$ for accounting purpose and to provide the visited network with some verifiable information to compose its accounting claims. Second, we address the traceability of the mobile users. In particular, it might be desirable to hide the user's identity from the visited network and achieve untraceability of its roaming sessions as it is currently the case in GSM/UMTS through the use of a Temporary Mobile Subscriber Identity (TMSI). Third, we want to protect $\mathcal{V}$ and $\mathcal{H}$ from denial-of-service attacks or, at least, mitigate their impact. The security goals related to accounting, untraceability and DoS protected are detailed respectively in Sections 6.4, 6.5 and 6.6 with regards to the basic SWISH proposal.

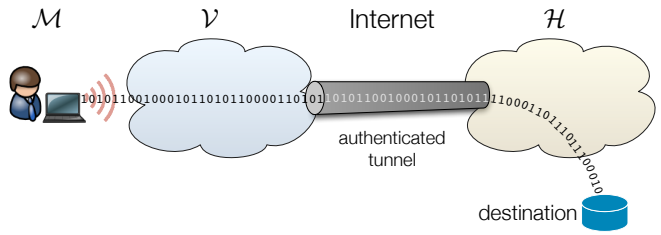
6.2 SWISH Overview

The SWISH approach is composed of two consecutive phases represented in Figure 6.1. First, a provably secure three-party *Roaming Authentication and Key Exchange (RAKE)* protocol is executed between $\mathcal{M}$, $\mathcal{H}$ and $\mathcal{V}$. Second, a tunnel is established to forward packets between the visited network and the home network throughout the roaming session.

RAKE provides authentication between all three entities participating in the roaming session, thus successfully thwarting various impersonation



(a) Phase 1: the RAKE protocol for authentication and key exchange between $\mathcal{M}$, $\mathcal{V}$ and $\mathcal{H}$



(b) Phase 2: the tunneling phase allows the mobile user to access to the Internet through his home network

Figure 6.1: The two phases of the SWISH mechanism

attacks. Furthermore, RAKE provides $\mathcal{M}$ and $\mathcal{H}$ with a session key K_{MH} for their secure end-to-end communication. In this way SWISH can ensure confidentiality of actual roaming data exchanged between $\mathcal{M}$ and $\mathcal{H}$ from curious $\mathcal{V}$ via symmetric encryption, which is realized in SWISH. Additionally, RAKE provides all three entities $\mathcal{M}$, $\mathcal{V}$, and $\mathcal{H}$ with an independent session key K_T. SWISH uses this key to protect the actual traffic forwarded by $\mathcal{V}$. Moreover, K_T can be used to derive other keys, including the WPA Master Session Key (MSK) if the user connects using a WiFi connection. K_T is also used by SWISH for secure negotiation of other roaming-specific parameters amongst the networks, such as those needed for accounting and billing. RAKE is also flexible in that it allows roaming participants to negotiate cryptographic algorithms for both tunnel authentication and encryption. A detailed specification of the actual RAKE protocol is provided in Section 6.3.

Once RAKE has succeeded, $\mathcal{M}$ holds an IP address from his home network and a tunnel between $\mathcal{V}$ and $\mathcal{H}$ is established (Figure 6.1b). This indicates the beginning of the tunneling phase. End-to-end encryption can be enabled between $\mathcal{M}$ and $\mathcal{H}$ to ensure confidentiality against $\mathcal{V}$. The $\mathcal{V}$-$\mathcal{H}$ tunnel is used by $\mathcal{V}$ to forward all packets exchanged between $\mathcal{M}$ and $\mathcal{H}$. In

contrast to VPN solutions [SSC07], the roaming tunnel to $\mathcal{H}$ is maintained by $\mathcal{V}$ and not by $\mathcal{M}$, which means that $\mathcal{M}$ does not control the way packets are forwarded through $\mathcal{V}$. This significantly reduces security risks for the infrastructure of the visited network. We provide more details about the tunnel establishment method in Chapter 7.

We decided to keep the encryption between $\mathcal{M}$ and $\mathcal{H}$ optional. If $\mathcal{V}$ is fully trusted by $\mathcal{M}$ and $\mathcal{H}$ and if confidentiality is not a requirement, this encryption should be disabled to prevent useless computational cost on both $\mathcal{M}$ and $\mathcal{H}$. Traffic impersonalization from an attacker on the Internet cannot be performed thanks to the authenticated tunnel between $\mathcal{V}$ and $\mathcal{H}$.

6.3 RAKE Protocol

The RAKE protocol is used for authentication and key exchange between $\mathcal{M}$, $\mathcal{H}$ and $\mathcal{V}$. To introduce this protocol, we first describe the protocol requirements and the cryptographic primitives used. Then we present a first high-level version of the protocol that achieves these goals. We then explain how keys are derived from shared information. Finally, we describe the complete protocol with all technical details needed to make it work in real environment. More details about our implementation are provided in Chapter 7.

6.3.1 Requirements and Constraints

The design of the RAKE protocol mostly relies on the security requirements described in Section 6.1 and hereafter. In addition to these, we introduce performance requirements. First, as the mobile device is typically a light device running on battery, we want $\mathcal{M}$ to perform as few computations as possible. Second, to complete the protocol in the shortest time, the amount of messages exchanged between $\mathcal{H}$ and $\mathcal{V}$, i.e., across the Internet, must be kept to a minimum. Besides that, the protocol should be flexible enough to support meaningful combinations of authentication and encryption amongst the participants during the roaming phase.

Trust Assumptions

The mobile device $\mathcal{M}$ and its home network $\mathcal{H}$ are assumed to maintain some security association (as a result of the initialization), and to accept the provided tunnel connection if they can successfully authenticate each other upon the tunnel establishment.

To the contrary, there is no security association between $\mathcal{M}$ and $\mathcal{V}$ prior to the execution of RAKE. This is natural, since assuming that networks are aware of mobile devices hosted by other networks and assuming that mobile devices are aware of networks that have roaming agreements with their home networks would obviously lead to scalability issues and stand in contrast with the actual roaming goal. The lack of trust between $\mathcal{M}$ and $\mathcal{V}$ results in several problems. First, $\mathcal{M}$ cannot rely on $\mathcal{V}$ in questions related to the authentication of $\mathcal{H}$, i.e., a malicious $\mathcal{V}$ may try to impersonate $\mathcal{H}$ towards $\mathcal{M}$. Second, $\mathcal{M}$ must be assured that $\mathcal{V}$ is authorized by $\mathcal{H}$ to create such tunnels. Note that, although this requirement seems not of prime importance for the end-to-end communication between $\mathcal{M}$ and $\mathcal{H}$ it provides additional robustness in the sense that the established tunnel will likely be maintained until either $\mathcal{M}$ or $\mathcal{H}$ decide to disconnect (as imposed by the roaming contract), thus lowering the risk that some unauthorized visited network decides to establish the tunnel connection and then spontaneously closes it. Third, $\mathcal{V}$ must be able to check that $\mathcal{M}$ is authorized by $\mathcal{H}$ to request tunnels to $\mathcal{H}$. This requirement is of importance in commercial roaming scenarios where $\mathcal{V}$ may charge $\mathcal{H}$ for the provided roaming service.

On the other hand, as part of their contract we assume that $\mathcal{V}$ creates a tunnel to $\mathcal{H}$ if it successfully authenticates $\mathcal{H}$, whereas $\mathcal{H}$ accepts the provided tunnel after the successful authentication of $\mathcal{V}$ (in addition to the authentication of $\mathcal{M}$). In particular, we assume that $\mathcal{V}$ creates a tunnel to $\mathcal{H}$ if it successfully authenticates $\mathcal{H}$ as the intended tunnel end point and receives the assurance that $\mathcal{M}$ is authorized to use the tunnel. Similarly, $\mathcal{H}$ accepts the provided tunnel only if it has been created by $\mathcal{V}$. On the other hand, we still assume that $\mathcal{V}$ can misbehave and try to impersonate $\mathcal{M}$ towards $\mathcal{H}$. There are several reasons for this assumption. First, in commercial roaming scenarios malicious $\mathcal{V}$ could try to establish tunnels to $\mathcal{H}$ at will and charge $\mathcal{H}$ subsequently. Second, a successful impersonation of $\mathcal{M}$ followed by the acceptance of the established tunnel by $\mathcal{H}$ could be

misused by $\mathcal{V}$ (e.g., to use the established tunnel to perform illegal activities in the name of $\mathcal{M}$).

Main Goals

Based on these assumptions, the RAKE protocol aims at fulfilling the two following goals.

To complete the RAKE protocol with success, $\mathcal{M}$ must have authenticated $\mathcal{H}$ as beting its home network, identified as id_H, and $\mathcal{H}$ must have authenticated $\mathcal{M}$ as being id_M, one of its mobile users. $\mathcal{H}$ (resp. $\mathcal{V}$) have also to authenticate $\mathcal{V}$ (resp. $\mathcal{H}$) as id_V (resp. id_H), one of its roaming partner.

A second objective is to derive two new session keys. The first key K_{MH} can only be derived by $\mathcal{M}$ and $\mathcal{H}$. The second one, K_T can only be derived by $\mathcal{M}$, $\mathcal{V}$ and $\mathcal{H}$.

6.3.2 Cryptographic Primitives

SWISH uses several well-known cryptographic primitives:

- A *pseudo-random function* $\texttt{PRF} : \{0,1\}^\kappa \times \{0,1\}^* \rightarrow \{0,1\}^*$ which is used for the purpose of key derivation and can be realized using block-ciphers or keyed one-way hash functions. No efficient algorithm can distinguish (with significant advantage) between a function chosen randomly from the PRF family and a random oracle. In the notation $\texttt{PRF}_k(l,x)$, k is the key, l is a publicly fixed label, and x is the input.
- An *asymmetric encryption scheme* satisfying the property of indistinguishability under (adaptive) chosen-ciphertext attacks (IND-CCA2) [RS99] whose encryption and decryption operations are denoted `Enc` and `Dec`, respectively.
- A *digital signature scheme* which provides existential unforgeability under chosen message attacks (EUF-CMA) whose signing and verification operations are denoted `Sig` and `Ver`, respectively.
- A *message authentication code* function `MAC` that satisfies *weak unforgeability against chosen message attacks* (WUF-CMA) [BN00], e.g., the

popular function HMAC [BCK96,Bel06] can be used for this purpose. In our protocol, the data is prepended with an integer to discriminate distinct usages of the function with the same key on the same data.

6.3.3 Overview

Figure 6.2 depicts the basic sequence of messages and actions of the RAKE protocol. Before starting the protocol, $\mathcal{M}$ and $\mathcal{H}$ share two secrets, k_M^{PRF} and k_M^{MAC}. The visited network has two key pairs, one for signing (sk_V, vk_V) (respectively the signing and verification keys) and one for encryption (dk_V, ek_V) (respectively the decryption and encryption keys). The home network also needs a signing key pair. For now, we simply suppose that $\mathcal{V}$ and $\mathcal{H}$ have previously exchanged their public keys securely. Each entity has an identifier, denoted *id*, which can be used as key to retrieve information about the already-known entities.

The RAKE protocol is composed of seven messages, named *I1*, *I2*, *I3*, *HA1*, *HA2*, *MA1* and *MA2*. The following does not describe the protocol in a chronologically order but details how the aforementioned security objectives are applied. Most security functions are applied on a session id, denoted *sid*, which is computed as the concatenation of all ids (id_M, id_V and id_H) and nonces generated by each entity (n_M, n_V and n_H). These nonces are generated randomly by each entity. They guarantee that each session id is different from the previous ones and so, that cryptographic values computed previously are not valid anymore. The identities and nonces are exchanged between entities and thus known by each of them.

The authentication between $\mathcal{H}$ and $\mathcal{M}$ relies on symmetric cryptography. The digital signatures μ_H and μ_M are computed using a MAC function on the session id, concatenated with a *zero* or a *one* to differentiate the values. The key that is used for this function is k_M^{MAC} which is only known from $\mathcal{M}$ and $\mathcal{H}$. Only these two entities are able to generate and check these values.

The authentication between $\mathcal{H}$ and $\mathcal{V}$ is based on asymmetric signatures, σ_H and σ_V, in *HA1* and *MA2* messages. These signatures are both computed on the session id. σ_H is also computed on χ (see below) to protect the integrity of this value.

The second security goal is to exchange keys that can be used later on.

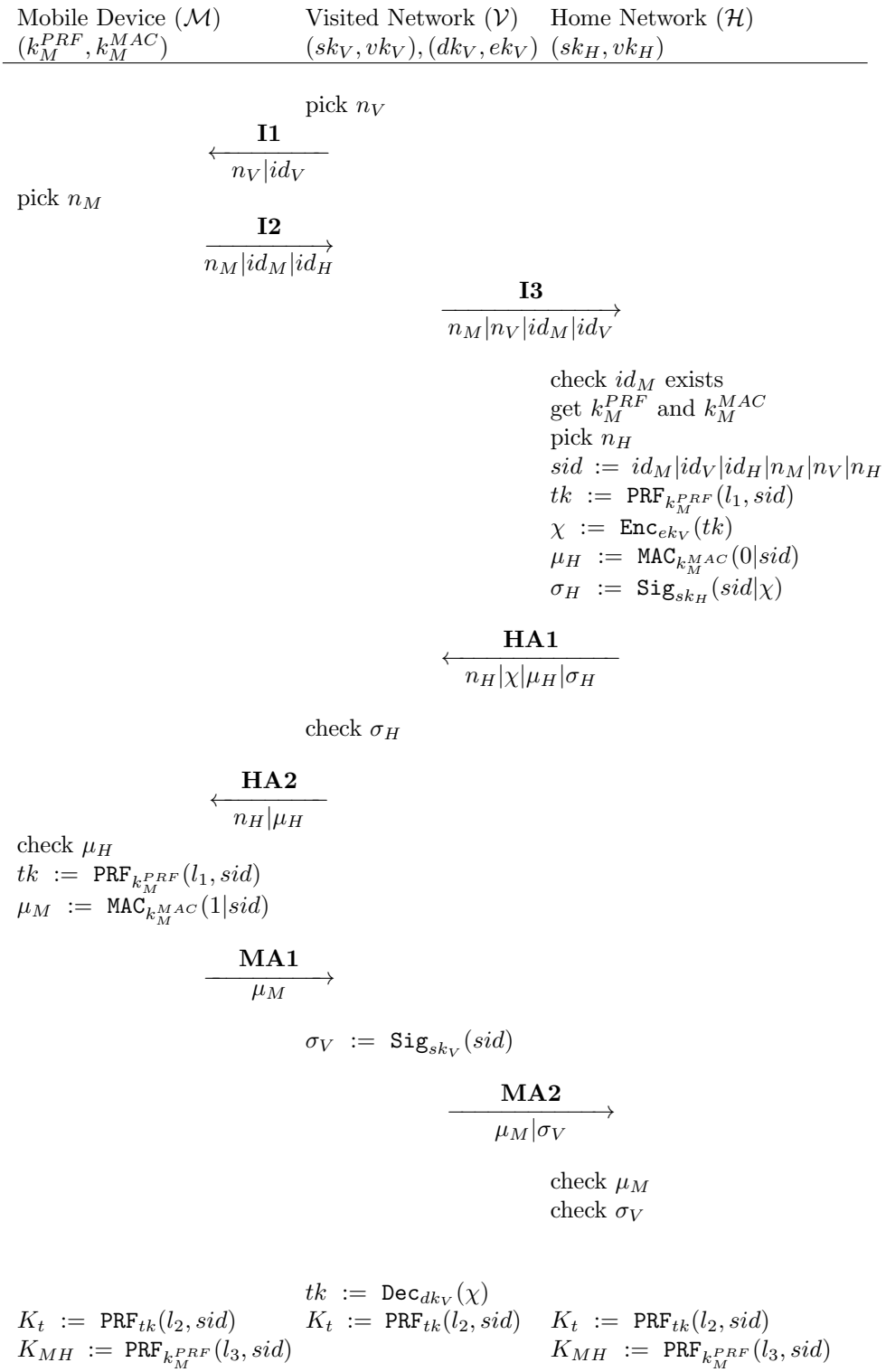


Figure 6.2: Overview of the RAKE protocol. The message names are indicated in bold above the arrows.

These keys are K_T, that is shared between the three parties, and K_{MH} that must only be known by $\mathcal{M}$ and $\mathcal{H}$. K_{MH} is directly derived from the session id using a pseudo-random function (PRF) and the key k_M^{PRF}. K_T is computed in the same way but by using the key tk. This temporary key (tk) is computed using a PRF and k_M^{PRF} as a key, and is sent encrypted (χ) to $\mathcal{V}$. The key derivation is detailed in Section 6.3.4.

More details on how actual payloads are built are given in Section 6.3.4. We proved in [MLK+08] that RAKE satisfies the previously defined security goals [1].

6.3.4 Key management and derivation

The authentication between $\mathcal{H}$ and $\mathcal{V}$ is performed using public key certificates. RAKE supports three different modes for the distribution of public keys. First, RAKE can use certificates that have already been pre-shared between the partner networks. However, this solution cannot be applied at a wide scale. Second, RAKE can obtain certificates from a centralized service that stores all the certificates and the corresponding revocation list (CRL), i.e., each time a network needs a certificate of another network it sends the corresponding request to the service. Third, RAKE allows networks to distribute their certificates during the protocol execution and provides the recipients with the ability to check the validity of the received certificates by comparing them to CRLs obtained from the central service. This last mode justifies to exchange certificates between entities, that is why this data is optional in the full protocol, shown in Figure 6.4.

$\mathcal{M}$ and $\mathcal{H}$ share a secret (e.g., a passphrase) from which all the keys are derived. The key derivation scheme is shown on Figure 6.3. K_{MH} and K_T are divided in substrings depending on the need for sub-keys. K_{MH} is used for K_{MH}^{MAC} (for accounting) and keys that may be needed for tunnel encryption. K_T is used for the EAP key, i.e for WiFi encryption, and for tunnel authentication.

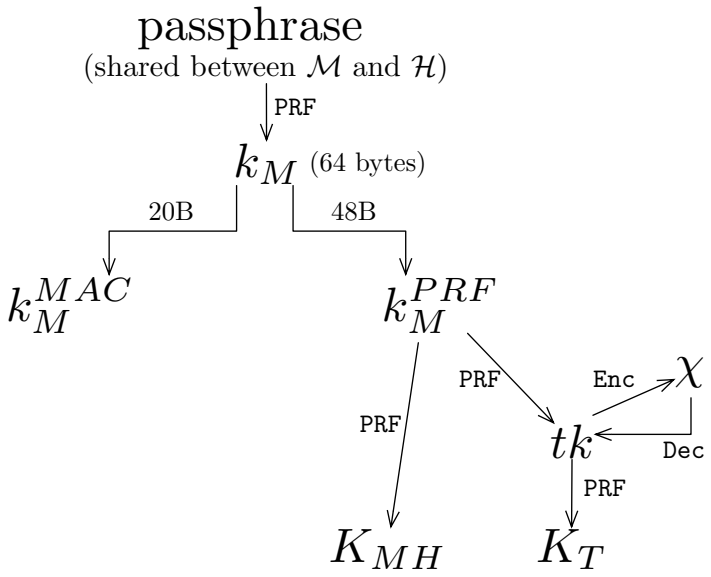


Figure 6.3: Key derivation in RAKE and SWISH

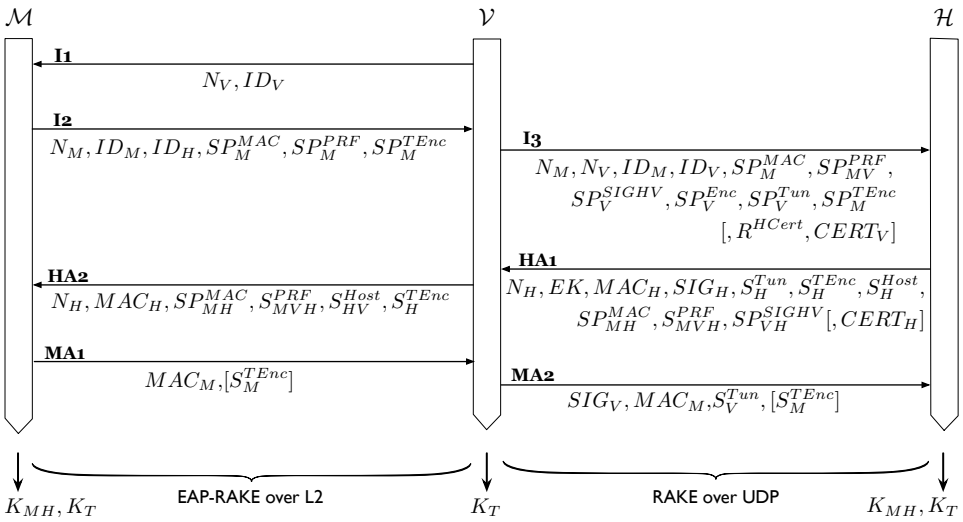


Figure 6.4: Full RAKE protocol including all optional payloads

6.3.5 RAKE payloads

Figure 6.4 depicts the full RAKE protocol, i.e., including all settings and practical aspects. Each RAKE message is encoded as a linked list of payloads. Each payload carries one piece of information with some parameters,

[1] [MLK+08] uses completely different notations from this thesis. The major ones are that SWISH and RAKE are respectively called WRT and `AWRT`.

e.g., the format of this data. The format of RAKE payloads is detailed in our technical report[2].

Security Proposal payloads have been added in Figure 6.4. These are used to negotiate which algorithm should be used, mainly for cryptographic primitives. The first entity sends its own security proposal containing all the algorithms it supports for a specific use, e.g., for the PRF function, for MAC computation, ... The receiver computes the intersection between this list and its own list. If the third party is implied, the updated security proposal is sent to this entity which performs the same operation.

The following describes each message and the payloads it contains.

I1 message (from $\mathcal{V}$ to $\mathcal{M}$)

N_V A nonce, randomly chosen by $\mathcal{V}$. The nonce size must be 32 bytes.

ID_V The identity of $\mathcal{V}$. The ID can either be an IP address, a fully-qualified name or an ASCII string.

I2 message (from $\mathcal{M}$ to $\mathcal{V}$)

N_M A nonce, chosen by $\mathcal{M}$. The nonce size must be 32 bytes.

ID_M The identity of $\mathcal{M}$. The ID can either be an IP address, a fully-qualified name or an ASCII string.

ID_H The identity of $\mathcal{H}$. The ID can either be an IP address, a fully-qualified name or an ASCII string.

SP_M^{MAC} The list of algorithms supported by $\mathcal{M}$ for the `MAC` function.

SP_M^{PRF} The list of algorithms supported by $\mathcal{M}$ for the `PRF` function.

SP_M^{TEnc} The list of algorithms supported by $\mathcal{M}$ for the tunnel encryption, including *none*.

[2]Can be downloaded from http://inl.info.ucl.ac.be/techrep-rake.

I3 message (from $\mathcal{V}$ to $\mathcal{H}$)

N_M	The nonce chosen by $\mathcal{M}$.
N_V	The nonce chosen by $\mathcal{V}$.
ID_M	The identity of $\mathcal{M}$.
ID_V	The identity of $\mathcal{V}$.
SP_M^{MAC}	The list of algorithms supported by $\mathcal{M}$ for the `MAC` function.
SP_{MV}^{PRF}	The list of algorithms supported by both $\mathcal{M}$ and $\mathcal{V}$ for the `PRF` function.
SP_V^{SIGHV}	The list of algorithms supported by $\mathcal{V}$ for the digital signature between $\mathcal{H}$ and $\mathcal{V}$ (σ_H and σ_V).
SP_V^{Enc}	The list of algorithms supported by $\mathcal{V}$ for asymmetric encryption (for χ).
SP_V^{Tun}	The list of protocols supported by $\mathcal{V}$ for the tunnel and its authentication.
SP_M^{TEnc}	The list of algorithms supported by $\mathcal{M}$ for the tunnel encryption.
R^{HCert}	(optional) If certificates are managed with up-to-date CRL (second scenario in Section 6.3.4), this requests $\mathcal{H}$ to send its certificate.
$CERT_V$	(optional) If certificates are managed with CRL (second scenario in Section 6.3.4), the certificate of $\mathcal{V}$.

HA1 message (from $\mathcal{V}$ to $\mathcal{H}$)

N_H	A nonce, chosen by $\mathcal{H}$. The nonce size must be 32 bytes.
EK	A payload containing χ which is tk encrypted for $\mathcal{V}$.
MAC_H	A payload containing μ_H.
SIG_H	A payload containing σ_H.

S_H^{Tun} The settings for $\mathcal{H}$-$\mathcal{V}$ tunnel: the protocol, the authentication mechanism and other settings depending on the type of tunnel.

S_H^{TEnc} The settings for $\mathcal{H}$-$\mathcal{M}$ encryption: the protocol, the cryptographic algorithms and other settings depending on the type of encryption. Can be *none* if both $\mathcal{M}$ and $\mathcal{H}$ prefers not enabling it.

S_H^{Host} The connection settings for the mobile, it can contain the IP address, the DNS server to use and the gateway.

SP_{MH}^{MAC} The list of algorithms supported by $\mathcal{M}$ and $\mathcal{H}$ for the `MAC` function.

SP_{MVH}^{PRF} The list of algorithms supported by $\mathcal{M}$, $\mathcal{V}$ and $\mathcal{H}$ for the `PRF` function.

SP_{VH}^{SIGHV} The list of algorithms supported by $\mathcal{V}$ and $\mathcal{H}$ for the digital signature between $\mathcal{H}$ and $\mathcal{V}$ (σ_H and σ_V).

$CERT_H$ (optional) If it has been requested by $\mathcal{V}$, the certificate of $\mathcal{H}$.

HA2 message (from $\mathcal{V}$ to $\mathcal{M}$)

N_H The nonce chosen by $\mathcal{H}$.

MAC_H A payload containing μ_H.

SP_{MH}^{MAC} The list of algorithms supported by $\mathcal{M}$ and $\mathcal{H}$ for the `MAC` function.

SP_{MVH}^{PRF} One `PRF` function algorithm supported by $\mathcal{M}$, $\mathcal{V}$ and $\mathcal{H}$, and used for all PRF computations.

S_{HV}^{Host} The connection settings for the mobile, it can contain the IP address, the DNS server to use and the gateway.

S_H^{TEnc} The settings for $\mathcal{H}$-$\mathcal{M}$ encryption: the protocol, the cryptographic algorithms and other settings depending on the type of encryption. Can be *none* if both $\mathcal{M}$ and $\mathcal{H}$ prefers not enabling it.

MA1 message (from $\mathcal{M}$ to $\mathcal{V}$)

MAC_M	A payload containing μ_M.
S_M^{TEnc}	(optional) The settings for $\mathcal{M}$-$\mathcal{H}$ encryption: the protocol, the cryptographic algorithms and other settings depending on the type of encryption.

MA2 message (from $\mathcal{V}$ to $\mathcal{H}$)

SIG_V	A payload containing σ_V.
MAC_M	A payload containing μ_M.
S_V^{Tun}	The settings for $\mathcal{V}$-$\mathcal{H}$ tunnel: the protocol, the authentication mechanism and other settings depending on the type of tunnel.
S_M^{TEnc}	(optional) The settings for $\mathcal{M}$-$\mathcal{H}$ encryption: the protocol, the cryptographic algorithms and other settings depending on the type of encryption.

6.4 An Adaptive Accounting Protocol

In commercial scenarios, Internet sharing may require both monitoring and control of the resource consumption, e.g., the available communication bandwidth, and an accounting process for the consumed resources. In existing sharing solutions the actual consumption of resources (time and/or volume) by mobile devices is measured solely by the visited network. These measurements when included in the bill can often no longer be verified by the end-users. As discussed in Section 5.1, the missing trust relationship between users and — possibly unknown — visited network providers imposes risks of overcharging that may remain unnoticed. Another problem is that current solutions based on a post-paid basis do not provide protection against repudiation of the later bills by the users. For this reason, many Internet sharing providers prefer the pre-paid method, which is, however, less usable and secure from the perspective of users. The architecture of SWISH allows for an elegant post-paid solution that thwarts the aforementioned risks. It

is the continuous presence of the home network throughout the roaming session that allows for the independent real-time measurement of the visited network's resources consumed by mobile users. The idea of SWISH accounting is thus to use the available trust between both networks, and between the mobile user and his home network, to minimize the accounting risks for the involved parties.

In this section, we propose a lightweight accounting protocol in which $\mathcal{H}$ can be charged for the forwarding service provided by $\mathcal{V}$ based on the number of bytes sent through the tunnel. The protocol limits the bill repudiation risk for $\mathcal{V}$ and can be deployed in a realistic environment where packet losses are frequent, available bandwidth likely to be high and where no trusted broker is available. The SWISH accounting protocol combines non-repudiable billing mechanisms with adaptive bandwidth allocation. This can be done efficiently since both networks are active during the entire roaming session. The main idea behind the accounting protocol is to use iterations. At the end of an iteration, $\mathcal{H}$ signs a receipt for the amount of data forwarded by $\mathcal{V}$ during this iteration. This allocated bandwidth is iteratively increased by $\mathcal{V}$ according to the number of receipts it has received. This growth is similar to the TCP *slow-start* approach. It ensures that no party can be cheated on more than a few bytes.

We remark that the SWISH accounting protocol is of independent interest for commercial communication scenarios in which some third party relays messages, e.g., it can be also used to charge two communicating clients through a SIP proxy. The protocol could also apply to other measurement units than a quantity of bytes, e.g., to the time.

6.4.1 Security Model for Accounting

We suppose that $\mathcal{M}$ and $\mathcal{H}$ trust each other and want to use the tunnel connection provided by $\mathcal{V}$ without being overcharged or suspended by $\mathcal{V}$ for not agreeing with the correct accounting information. On the other hand, $\mathcal{V}$ wants the ability to obtain non-repudiable confirmation from $\mathcal{H}$ for the consumed resources for charging purpose.

However, packets sent across the Internet may be lost and it is intrinsically impossible to distinguish packets that have been lost from packets that

have been deliberately dropped. The problem is that $\mathcal{V}$ may try to drop packets while claiming their correct transmission. Likewise, $\mathcal{H}$ may claim that no packets have been received although their delivery did take place.

The SWISH accounting protocol has two security objectives. First, to protect the mobile device $\mathcal{M}$ and the home network $\mathcal{H}$ from a dishonest visited network $\mathcal{V}$ trying to overcharge $\mathcal{H}$ for more bytes than actually forwarded by $\mathcal{V}$ between $\mathcal{H}$ and $\mathcal{M}$. The second objective is to protect $\mathcal{V}$ from colluding $\mathcal{M}$ and $\mathcal{H}$ that aim to force $\mathcal{V}$ to transmit more bytes than $\mathcal{V}$ agreed. Additionally, we require that if one party disconnects or claims that the other one has violated the agreement, the data volume that cannot be charged remains below some threshold. These objectives are formalized in the following definitions.

We define the *acceptable loss ratio* and the *acceptable uncharged ratio* as follows. The *acceptable loss ratio*, $\rho_{\mathcal{H}}$, is the value such that, for any $i \geq 0$, $\mathcal{H}$ will accept to sign receipts for Q_i bytes when $\mathcal{H}$ can check that at least $(1 - \rho_{\mathcal{H}})\, Q_i$ bytes have been actually forwarded by $\mathcal{V}$. The *acceptable uncharged ratio*, $\rho_{\mathcal{V}}$, is the value such that, for any $i \geq 0$, $\mathcal{V}$ will accept the risk to terminate a session having forwarded a total of $(1 + \rho_{\mathcal{V}})\, Q_i$ bytes while having obtained a receipt for only Q_i bytes.

Note that in the definition of non-repudiation we tolerate when $\mathcal{V}$ transmits up to $\rho_{\mathcal{V}}\, Q_i$ bytes without obtaining a receipt for this amount. The reason for such an optimistic approach is that, in asymmetric Internet communication, no fairness of accounting can be ensured without relying on some trusted broker or costly hardware. On the other hand, $\mathcal{V}$ provides the tunnel service, and therefore, it is meaningful to assume that at most $\rho_{\mathcal{V}}\, Q_i$ forwarded bytes are offered for free.

6.4.2 The Non-Repudiable Accounting Protocol

We first detail an iteration of the protocol, then explain how the iterations can overlap to increase performance and how the adaptive bandwidth allocation works.

Basic Iteration

Figure 6.5 shows, above each box, the values known by each entity after the RAKE execution. $sk_{\mathcal{H}}$ and $vk_{\mathcal{H}}$ are respectively $\mathcal{H}$'s signing and verification keys. We assume that $\mathcal{M}$ and $\mathcal{H}$ have established an authenticated tunnel so that $\mathcal{V}$ is not able to modify the packets nor inject forged packets.

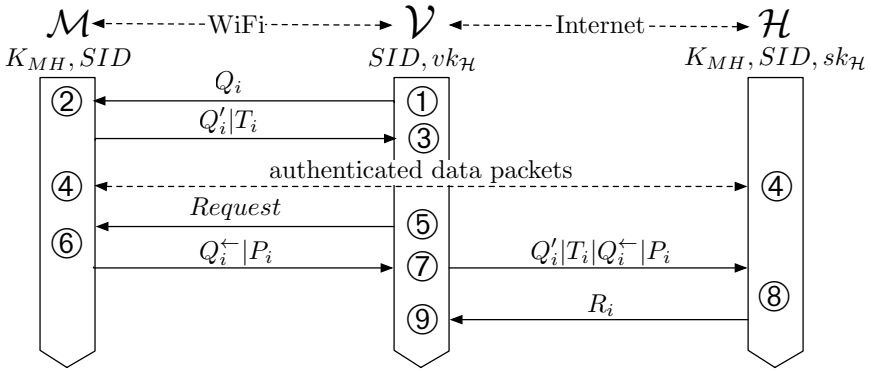


Figure 6.5: An iteration of the accounting protocol

Before each iteration, $\mathcal{M}$ negotiates with $\mathcal{V}$ the amount of data it is allowed to send and receive. We describe below the messages exchanged during one iteration of the protocol as depicted in Figure 6.5. We define q_i as the amount of bytes allocated to the current transmission "chunk", and Q_i as the total amount of bytes allocated during a set of iterations, i.e., $Q_i = \sum_{j=0}^{i} q_j$. This use of cumulative Q_i reduces the storage need, as the latest receipt can be considered as an ultimate proof.

① Upon reception of a receipt for the previous iteration, $\mathcal{V}$ chooses q_i, which is the new amount of bytes it agrees to forward during this iteration. It computes Q_i — i.e., the cumulative number of bytes it has allowed until now — and sends this value to $\mathcal{M}$.

② $\mathcal{M}$ chooses q'_i, the number of bytes it agrees to buy for the current iteration, so that $Q'_i \leq Q_i$ (or $q'_i \leq q_i$). It sends to $\mathcal{V}$ the value Q'_i and a ticket T_i proving this commitment. The ticket T_i is computed using a Message Authentication Code (MAC) on input Q'_i and SID with key K_{MH}^{MAC}. It is used as a proof for $\mathcal{H}$ that $\mathcal{M}$ agrees on the quantity. In most cases, the mobile will choose $q'_i = q_i$ which allows obtaining the highest bandwidth.

③ $\mathcal{V}$ stores the received values $Q_i'|T_i$ and starts forwarding packets. During this transmission $\mathcal{V}$ measures $Q_i^{\mathcal{V}}$, which is the amount of data forwarded in both directions .

④ $\mathcal{M}$ starts sending data. $\mathcal{M}$ and $\mathcal{H}$ count the amount of data they receive denoted $Q_i^{\leftarrow}$ and $Q_i^{\rightarrow}$, respectively.

⑤ After transmission of $Q_i^{\mathcal{V}} \geq Q_i'$ bytes, $\mathcal{V}$ requests $\mathcal{M}$ to provide the exact received volume and the proof.

⑥ In response, $\mathcal{M}$ sends $Q_i^{\leftarrow}|P_i$, where P_i is a proof for $\mathcal{H}$ that $\mathcal{M}$ actually received $Q_i^{\leftarrow}$ bytes of data, i.e., a MAC on the quantity and SID.

⑦ $\mathcal{V}$ forwards Q_i', $\mathcal{M}$'s ticket T_i, $Q_i^{\leftarrow}$ and proof P_i to $\mathcal{H}$.

⑧ $\mathcal{H}$ first checks whether the loss ratio is acceptable, i.e., if $\frac{Q_i' - (Q_i^{\rightarrow} + Q_i^{\leftarrow})}{Q_i'} \leq \rho_{\mathcal{H}}$ then $\mathcal{H}$ sends a non-repudiable receipt R_i back to $\mathcal{V}$. This receipt is computed as a signature using $sk_{\mathcal{H}}$ on Q_i' and SID. Otherwise, $\mathcal{H}$ disconnects without validating this iteration of the protocol. Note that $\rho_{\mathcal{H}}$ should not be fixed to a too low value to avoid abrupt disconnections. Recent studies have shown that the packet loss ratio on web pages across the Internet is about 1 to 1.5 percent [Bel10].

⑨ $\mathcal{V}$ always stores the latest tuple (R_i, Q_i', SID) it received. This tuple serves as a non-repudiable proof that $\mathcal{M}$ communicated at least Q_i' bytes.

Combining Iterations

In order to avoid traffic blocking between steps ⑤ and ⑨, $\mathcal{V}$ sends in advance the Q_{i+1} message in step ⑤. This means for $\mathcal{V}$ that it can be cheated on for $q_i + q_{i+1}$ bytes in the worst case. This happens if $\mathcal{H}$ does not send R_i before the end of the $(i+1)^{\text{th}}$ iteration.

Adaptive Bandwidth Allocation

We now describe how q_i values are computed. A too low q_i would bind the bandwidth and increase the frequency with which signatures have to be computed and verified; on the other hand, a too high q_i would increase the potential loss for $\mathcal{V}$ if $\mathcal{H}$ refuses to provide signed receipts.

As defined in Section 6.4.1, we want to ensure at iteration i that $\mathcal{V}$ is not cheated for more than $\rho_{\mathcal{V}} Q_{i-2}$. This can be done by using the following equation to compute q_i values:

$$q_i = \rho_{\mathcal{V}} Q_{i-2} - q_{i-1} \tag{6.1}$$

At startup, $\mathcal{V}$ only agrees to forward small amounts of data ($q_0 = q_1 = q_{min}$) before the first receipt has been received. When the latter has been validated, i.e., for $i > 2$, Equation (6.1) is used. We also force q_i to remain between acceptable bounds $q_{\min}$ and $q_{\max}$.

In [LDC+11] we prove the security of the accounting protocol, i.e., that it provides overcharge protection and non-repudiation of accounting.

6.4.3 Validation

In order to validate the accounting mechanism, we built a simple model of it into the ns-2 simulator [NS2]. Our main objective was to determine the impact of both $\rho_{\mathcal{V}}$ and the $\mathcal{H}$-$\mathcal{V}$ delay on the TCP throughput. We also wanted to detect eventual weird interactions between TCP and the accounting mechanism.

The protocol was implemented as a dedicated module on a router. This module models the behavior of $\mathcal{V}$: it keeps track of the number of bytes forwarded and blocks transmitted while waiting for tickets from $\mathcal{M}$ or receipt from $\mathcal{H}$. At each iteration, Q_i is incremented with q_i computed according to Equation (6.1).

To evaluate the impact on the performance of TCP, we used an environment with three nodes, corresponding to each party, connected with a 8 Mbps link. Our TCP source, attached to $\mathcal{M}$, uses the Linux TCP Implementation for ns-2. Our simulations indicate, as shown on Figure 6.6, that when the $\mathcal{V}$-$\mathcal{H}$ *Round-Trip Time* (RTT) is lower than 100 milliseconds, the accounting mechanism has no impact on the TCP throughput, even for very low values of $\rho_{\mathcal{V}}$ such as 0.1%. With longer RTT, such as 150 (respectively 200) milliseconds, the duration for the throughput maximization is 40 (respectively 63) seconds. Nevertheless we expect that in practice most utilizations of SWISH will experience a round-trip time smaller than 100 milliseconds which is a typical round-trip time between well connected nodes in

Europe and the U.S. For longer RTT, the impact on TCP throughput starts to be visible when $\rho_{\mathcal{V}}$ ratio is low. For example, Figure 6.7 shows that with $\rho_{\mathcal{V}} = 5\%$ and RTT=150 msec, the average TCP throughput during the first ten seconds after the authentication is about 90% of the TCP throughput achieved without accounting. This is still acceptable for most applications of WiFi sharing. To summarize, in most case, the TCP throughput will not be affected by the accounting protocol. Only networks with high-delay connections should not use too low values for $\rho_{\mathcal{V}}$ if they want to maximize the throughput in a few seconds.

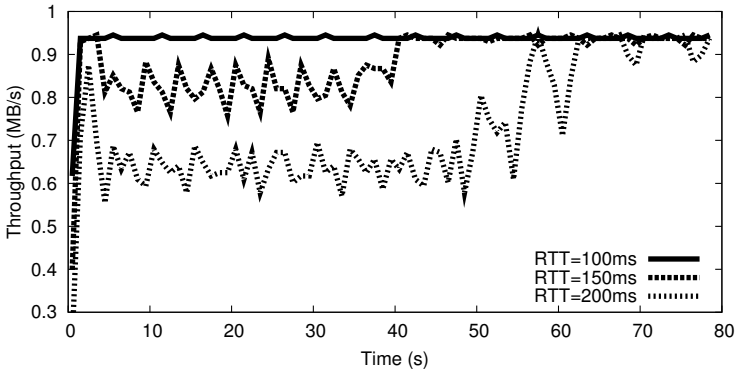


Figure 6.6: Impact of *Round-Trip Time* (RTT) on the TCP throughput ($\rho_{\mathcal{V}} = 0.001$). For low RTT, the accounting protocol has no impact on the TCP throughput. For RTT higher than 100ms, the TCP throughput is maximized after a few seconds.

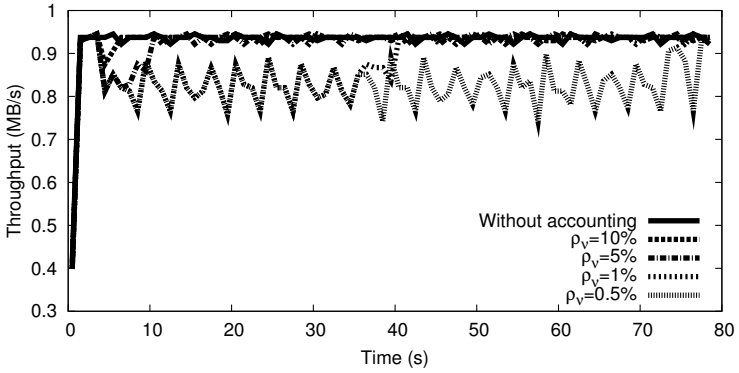


Figure 6.7: Impact of $\rho_{\mathcal{V}}$ on the TCP throughput when $RTT = 150ms$. If $\mathcal{V}$ chooses to reduce its risks (lower $\rho_{\mathcal{V}}$), the TCP throughput takes more time to be maximized.

6.5 Untraceability of Mobile Users

The RAKE protocol, as defined in Section 6.3, does not guarantee the anonymity of mobile users towards visited networks, thus enables user traceability by the latter. Indeed, the first message sent by $\mathcal{M}$ in RAKE protocol reveals its identity. In our context, we define the *untraceability* as the impossibility for a visited network to link with absolute certainty two visits as initiated by the same $\mathcal{M}$. The *anonymity* can be considered a sub-property as it is required but not sufficient to satisfy untraceability.

As discussed in Section 5.1.1, the lack of untraceability against visited networks may contradict the privacy demands of the users but is of importance for their identification in case of misbehavior. Existing roaming solutions do not have simultaneous support for user untraceability and misbehaving user tracing (see Table 5.1). To the contrary, the SWISH framework allows for an elegant solution to this problem due to the active involvement of the home network into the roaming process. At a high level, we encrypt the identity of the mobile user in the RAKE protocol, which can then be decrypted only by the home network. The use of *IND-CCA2* public-key encryption, which provides unlinkability between two executions on the same input, allows further to achieve untraceability among roaming sessions of the same user. At the same time, we preserve the ability to identify misbehaving users; identification of such users is now performed by the home network. In what follows, we specify the untraceability requirements for mobile users towards visited networks and specify an optional extension to the original RAKE protocol. The public-key encryption mechanism for user identities is realized in a special way that keeps the computational costs of $\mathcal{M}$ low aiming to avoid unnecessary public-key operations at the mobile device.

The basic idea of the SWISH untraceability extension to RAKE is that $\mathcal{M}$ sends a temporary pseudonym t_M (computed as the encryption of $\mathcal{M}$'s identity) to $\mathcal{H}$, instead of n_M in the *I2* message. At the end of the RAKE execution, $\mathcal{H}$ encrypts $\mathcal{M}$'s identity and transmits it to $\mathcal{M}$. This new pseudonym is used for the next execution of RAKE. In this way, linking two $\mathcal{M}$'s sessions is not possible for $\mathcal{V}$.

The protocol is designed to minimize the computational cost for $\mathcal{M}$. $\mathcal{M}$ will not have to compute the encryption itself, unless an attack occurs, i.e.,

one participant is dishonest. In that worst case, the computational cost will be higher for $\mathcal{M}$, as it needs to recompute itself the new temporary pseudonym, but the security remains unaffected.

The building blocks for this protocol are an *IND-CCA2* asymmetric encryption scheme $\texttt{Enc}/\texttt{Dec}$, a symmetric encryption scheme E/D and a message authentication scheme ($\texttt{MAC}$). We make the assumption that $\texttt{MAC}$ provides key privacy, meaning that an active attacker cannot decide which key was used to compute the MAC. This property is satisfied by MACs modeled as PRFs. In addition to the parameters of the basic RAKE protocol, the home network must use a public key ek_H and a corresponding private key dk_H for the asymmetric encryption scheme $\texttt{Enc}$. The mobile $\mathcal{M}$ should store ek_H and t_M, computed as $\texttt{Enc}_{ek_H}(\mathcal{M})$ (its identity encrypted under the public key of $\mathcal{H}$).

Mobile Device ($\mathcal{M}$)		Home Network ($\mathcal{H}$)
K_{MH}		$K_{MH}, (dk_H, ek_H)$
retrieve t_M or compute $t_M := \texttt{Enc}_{ek_H}(\mathcal{M})$	$\xrightarrow[t_M]{}$	
		$\mathcal{M} := \texttt{Dec}_{dk_H}(t_M)$ check $\mathcal{M}$ $t'_M := \texttt{Enc}_{ek_H}(\mathcal{M})$ $\psi := E_{K_{MH}}(t'_M)$ $\xi := \texttt{MAC}_{K_{MH}}(\psi)$
	$\xleftarrow[\psi \mid \xi]{}$	
verify ξ $t_M := D(\psi)$		

Figure 6.8: The SWISH anonymity extension to RAKE prevents the traceability of the mobile users by the visited networks.

The SWISH untraceability extension to RAKE, depicted on Figure 6.8, is integrated into the original protocol and works as follows:

1. In the first message (*I1*) of the RAKE protocol, $\mathcal{M}$ sends t_M instead of sending its identity $\mathcal{M}$.
2. $\mathcal{H}$ decrypts the message and obtains the identity of $\mathcal{M}$. In addition to K_{MH} computed in the original RAKE protocol $\mathcal{H}$ computes a new

pseudonym for $\mathcal{M}$ as $t'_M := \mathcal{E}_{pk_{\mathcal{H}}}(\mathcal{M})$, and sends using an Encrypt-then-MAC approach [BN08]: $E_{K_{MH}}(t'_M), \mathtt{MAC}_{K_{MH}}(E_{K_{MH}}(t'_M))$.

3. Upon receiving this message, $\mathcal{M}$ uses K_{MH} to obtain t'_M and checks the MAC value. If the MAC value is correct, $\mathcal{M}$ stores its new pseudonym $t_M := t'_M$. Otherwise, it computes t_M itself.

With this extension, $\mathcal{M}$ sends a different t_M at each RAKE execution. Thanks to the IND-CCA2 property of $\mathtt{Enc}$, each execution of $\mathtt{Enc}_{ek_H}$ on the same plaintext $\mathcal{M}$ results with overwhelming probability in a different ciphertext.

We remark that we do not consider the security proposal payloads as compromising untraceability. Even if a user is likely to keep its security settings unchanged from one RAKE execution to another one, such settings are usually set up by the application or by the operating system, resulting in a high collision rate with other users. Anyway, observations on security proposal values will never result in absolute certitude that two different executions of the RAKE protocol have been performed by the same or by different users.

6.6 Protection against Denial-of-Service

Here we present some ideas on how to enhance RAKE towards resistance against some types of DoS attacks. DoS attacks include all kind of attacks which result in service disruption for some period. However, it is not possible to protect a network architecture against all kind of DoS. For instance, preventing a user from connecting on WiFi is easy by using a WiFi jammer. In the same way, using a Distributed DoS attacks to isolate $\mathcal{V}$ or $\mathcal{H}$ from the Internet are out of our scope. The only DoS attacks we want to mitigate are attacks from a single anonymous user on the Internet or a mobile device, located in $\mathcal{V}$, that would cause $\mathcal{H}$ or $\mathcal{V}$ to perform more computations than the ones required for sending the attack.

Due to the higher communication delays on the path between $\mathcal{V}$ and $\mathcal{H}$ it might be desirable to decrease the risk that $\mathcal{V}$ opens a connection to $\mathcal{H}$ for the third protocol message without gaining stronger confidence that the party which requested the tunnel is a valid mobile device hosted by $\mathcal{H}$;

otherwise a DoS-attacker can compose the second protocol message, *I2*, and then simply close its own connection. A possible solution to minimize this risk is to equip $\mathcal{M}$ with a key pair and a certificate issued by $\mathcal{H}$, and demand a signature on this second message (which also includes a fresh nonce of $\mathcal{V}$). Observe, that this solution, although computationally expensive, minimizes the risk since the attacker must either forge the signature or be a holder of some valid certificate. Nevertheless, it is not completely satisfactory since $\mathcal{V}$ is not able to judge (without further interaction) whether the certificate has not been revoked by $\mathcal{H}$. One could further reduce the risk by requiring that device certificates are issued for some short validity period. We remark that this solution is incompatible with the untraceability extension presented in Section 6.5 as the signature may reveal information about $\mathcal{M}$'s identity. In practice, this should be enabled by $\mathcal{V}$ if it observes some abuses. This requires to add in message *I1* a *Signature Request* payload, a *Security Proposal* payload for the signature, and an optional $\mathcal{H}$*'s Certificate Request* payload if the certificates are managed using CRL. In message *I2*, it requires to add payloads for the signature, for $\mathcal{M}$'s certificate and eventually for $\mathcal{H}$'s certificate.

A similar threat is given for $\mathcal{H}$ which could be forced to keep the connection to $\mathcal{V}$ after the message *HA1* without gaining stronger confidence that both parties $\mathcal{M}$ and $\mathcal{V}$ are legitimate; otherwise a DoS attacker may flood $\mathcal{H}$ with *I3* messages and close its own connection thereafter. The risk here can be minimized by requiring the *I3* message to be signed by $\mathcal{V}$ whereby a time-stamp would also serve as a protection against replay attacks. Note that this signature cannot replace σ_V from the last message as this is required for the mutual authentication between $\mathcal{V}$ and $\mathcal{H}$. Once again, this kind of protection should only be enabled, eventually automatically, when an abuse is detected in the same way as some IKE implementations.

Since the RAKE protocol is based on UDP between $\mathcal{V}$ and $\mathcal{H}$, a DoS attack could be mounted using fragmented IP packets. As explained by Kaufman *et al.* [KPS03], the protocols running on top of UDP that require sending large packets depend on IP packet reassembly. As it was done with IKE, one of the defense proposed [KPS03], depending on the deployment scenario, should be chosen to avoid such attacks to succeed.

6.7 Related Work

The initial solution for the tunnel-based WiFi roaming by Sastry *et al.* [SSC07] is based on VPN tunnels securing the end-to-end communication between the mobile device and its home network. In their scheme the visited network accepts every device without any authentication and grants it access to the home network over the Internet. The mobile device can thus initiate a VPN connection (using NAT traversal techniques if necessary). However, this solution has several weaknesses. First, the Internet access granted by the visited network, even a restricted one, may bear intrusion risks to its infrastructure. Second, the mobile device must comply with the network layer infrastructure of the visited network (e.g., IPv4/IPv6, IP assignment via DHCP). Third, VPN tunnels do not provide any proof to the visited network that the mobile device is connecting to its real home network as a VPN connection can be established to any server on the Internet. Fourth, visited and home networks do not authenticate each other, and as a consequence, neither accounting mechanisms nor quality-of-service contracts can be securely implemented. We fairly remark that Sastry *et al.* were focusing on the architecture for the city-wide WiFi roaming rather than dealing with the related authentication and key establishment goals.

Salgarelli *et al.* [SBG+03] suggested a general roaming authentication framework based on shared keys which can be implemented as an EAP method. Their protocol extends the Needham-Schroeder technique [NS78] to accommodate the authentication servers of the visited and the home network while minimizing the communication rounds between them. Previously, Molva *et al.* [MST94] described another roaming protocol based on shared keys, which was designed for the integration into the IBM's KryptoKnight authentication and key distribution framework. Merino *et al.* [MMS+05] proposed a Single Sign-On authentication architecture based on 802.1X and EAP-TLS [SAH08] relying on the Public Key Infrastructure (PKI). Their method can be combined with any web-based authentication method, e.g., UAM. The drawback of this approach is that the mobile device is assumed to be able to check the validity of the visited network's certificate while being off-line. Furthermore, the use of public-key operations might be costly for performance-constraint mobile devices. Similar drawbacks appear in the authentication protocols from [GPS+03, BEG+]. The latter suggests

to delegate the verification of visited network's certificate to a trusted server. Long *et al.* [LWI04] suggested a roaming protocol based on the modified SSL handshake assuming that mobile devices are equipped with public-key certificates, so that the protocol can be executed without active involvement of the home network. Ribeiro *et al.* [RSZ04] described a roaming authentication approach based on IPsec VPNs and a hierarchy of certification authorities. The aforementioned problems with validation of public-key certificates by the mobile device were solved by Meyer *et al.* [MCW05] via secret sharing technique [Sha79]. In their protocol described as an extension of EAP-TLS [SAH08] each visited network is assumed to hold a share of the home network's secret key and the respective public-key certificate of the home network is pre-installed at the mobile device. During the execution of the protocol (which is a modified TLS handshake) the visited and the home network need to cooperate in order to perform the required signature and decryption operations.

The aforementioned solutions proposed for wireless non-tunnel-based roaming (in mobile phone and wireless IP networks) have been designed with the main goal to authenticate (and provide a session key to) the mobile and the visited network, whereby some approaches require interaction with the home network.

Much work has been done on the topic of fair non-repudiation of exchange, in which no party gains any advantage over other parties [KMZ02]. These protocols are mainly based on a trusted third-party and do not scale well to a real Internet environment. Hasan *et al.* presented a simpler roaming approach of non-repudiation [HS05] that can be applied in our architecture, but their algorithm is executed after each session, which increases the delay at each session ending as well as the loss of money in case of dispute. Goldberg *et al.* proposed a monitoring technique in presence of a man-in-the-middle that tries to bias measurements [GXT+08]. Their solution could be used in the accounting domain, but appears less satisfying than ours in terms of security and efficiency. The fact that our solution exhibits better performance than the aforementioned ones is not surprising if we consider that it is specifically tailored for a three party infrastructure, which induces strong hypotheses and permits us to relax the problem.

User privacy is a popular research topic and fits in a wide variety of

protocols. For example, secret handshakes (as initiated in [BDS+03]) allow two users to learn whether they are members of the *same* group only if they possess corresponding membership credentials for this group and without disclosing the membership information in the opposite case. However, existing privacy-preserving authentication protocols do not give a direct, ad-hoc solution for the SWISH protocol, thus motivating the design of our anonymity and untraceability extension.

Another family of related protocols are Mobile IP [Per02] and Mobile IPv6 [Sol04]. Mobile IP was designed to allow a mobile host to change its point of attachment without changing its IP address. This is achieved by tunneling the packets to and from the mobile through a Home Agent located in the home network of the mobile. There are several important differences between SWISH and the Mobile IP solutions. First, unlike Mobile IP, SWISH does not assume that a mobile node will automatically receive an IP address in any visited network. Given the security issues in today's Internet, providing an IP address to an unknown node might be risky. Second, the SWISH authentication mechanism involves the mobile node, the visited network and the home network while with Mobile IP the visited network does not participate in any authentication. More recently, Proxy Mobile IPv6 (PMIP) was proposed at IETF [GLD+08]. PMIP does not require the mobile device's participation by using a PMIP gateway (a proxy), in the visited network, which allows the mobile user to receive an IP address from his home network. The PMIP architecture is actually very similar to the SWISH one, especially in IPv4 scenarios [WG10]. The main difference is that PMIP allows route optimization under three following assumptions: the encryption between $\mathcal{M}$ and $\mathcal{H}$ is not required, IPv6 is used by all parties and the destination supports Mobile IPv6. Even if RFC5779 [KBC+10] defines the interaction between PMIP and the Diameter protocol [CLG+03], there is no specific authentication protocol dedicated to the PMIP architecture.

Most of the standard authentication and key establishment protocols for mobile phone networks, e.g., [3GP08, ETS08, RK04], are based on the pre-shared key between the home network and its mobile device. These protocols establish the session key between the mobile device and the foreign network. Several solutions have been further proposed to allow roaming among different mobile phone networks. For example, the authenticated

roaming between GSM and UMTS has been specified within the UMTS standard [3GP07] and partially addressed in [3GP08] and the roaming procedure between UMTS and CDMA2000 has been addressed in [KCH+03]. Notable is also the man-in-the-middle attack discovered by Meyer and Wetzel [MW04] by which the attacker can impersonate a GSM base station to a UMTS subscriber as a result of missing integrity protection in GSM.

The IP Multimedia Subsystem (IMS) is a technology that aims at merging the Internet with the cellular world. This is the key element in the 3G architecture that makes it possible to provide ubiquitous access to all Internet services. In roaming scenarios, the IMS gateway to the Internet can be either located in the visited or in the home network. The former is the long term vision of IMS but requires both operators to be 3GPP Release 5-compliant [CGM04]. The authentication and authorization of the user is done by using SIP `REGISTER` messages [RSC+02]. If the user is a visitor, his credentials can be retrieved using the *Diameter* protocol [CLG+03] between authentication servers. Once authenticated, the mobile device established IPsec ESP tunnels with a specific server in the visited or home network. In IMS, the charging architecture [3GP05, 3GP10] defines the offline and the online charging models, corresponding respectively to the post-paid and pre-paid models. In offline charing, nodes that report charging events can be either located in the visited or in the home network. These nodes sends accounting information to a Billing System (BS) using the *Diameter* protocol. BSs in different domains exchange information using nonstandard means [CGM04]. Between the mobile device and the network nodes, a specific 3GPP extension to the SIP protocol [GMHM03] is used. In online charging, services are paid by credit units; some credit units are used to directly pay a service, others are used to reserve a number of units to provide service to the user [3GP06, CLG+03]. Even if this latter form may look similar to our accounting protocol, it does not include any adaptive mechanism which prevents from losing money if the server did not provide the service it pretends to. [CGM04]

In parallel to our work, Heer *et al.* defined a WiFi sharing solution based on tunnels, called PiSA [HGWW08]. Their solution is also based on Sastry *et al.*'s proposal [SSC07] and relies on the same trust assumptions between $\mathcal{M}$, $\mathcal{V}$ and $\mathcal{H}$. PiSA extends HIP [MN06] to add end-to-middle

authentication and signaling. They modify the two-party authentication of HIP to accept a third party, the visited network's access point. The authentication phase of PiSA consists of four packets sent between $\mathcal{M}$ and $\mathcal{H}$, and unlike SWISH, requires $\mathcal{M}$ to perform asymmetric encryption. For the tunneling phase, $\mathcal{M}$ sends his data to $\mathcal{H}$, that acts as an HIP proxy, through an ESP tunnel. To ensure that the packets from the mobile device are only sent to his home network, the access point is configured to only authorize authenticated HIP flows. As an advantage, since $\mathcal{V}$ acts as a filter and does not perform tunneling, PiSA is a bit lighter than SWISH for $\mathcal{V}$ during the tunneling phase. As drawbacks, PiSA requires the mobile to use ESP for tunneling while, in SWISH, thanks to IEEE 802.11 (WiFi) hardware encryption, the ESP tunnel can be safely disabled as long as the AH tunnel is enabled between $\mathcal{V}$ and $\mathcal{H}$. Note that in that case, we are not protected against a malicious visited network. We also notice that HIP is not widely deployed on both end-hosts and middle-boxes. This may require deep changes and setting up an HIP address resolution server. Finally, in PiSA, the untraceability against $\mathcal{V}$ cannot really be ensured [HGWW08]. Anyway, PiSA mostly benefits from the same advantages as SWISH and suffers from the same drawbacks, i.e., higher delay, each party need to be modified, ...

Recently, Noack [Noa09] proposed a protocol presented as an improvement of RAKE. This protocol has a mechanism for mobile anonymity which relies, like ours, on one-time pseudonyms. A drawback of his solution is that it forces $\mathcal{H}$ to store a new one-time pseudonym after each session. Moreover, unlike our protocol, it does not satisfy unlinkability. The reason is that $\mathcal{M}$ and $\mathcal{H}$ compute the next one-time pseudonym at the end of the protocol using K_{MH}, so if $\mathcal{M}$ never receives any message after sending its first message, $\mathcal{M}$ will not be able to compute K_{MH}. Therefore, $\mathcal{V}$ just has to drop some packets from $\mathcal{M}$ to prevent $\mathcal{M}$ from connecting anonymously. Noack also proposed an accounting mechanism, but it is not adaptive and significantly increases the workload of $\mathcal{M}$, that has to verify public key signatures. This makes the protocol difficult to use in actual network environments using high bandwidth, where a good compromise between computational cost and loss risk might be difficult to achieve. Finally, we note that one of the methods claimed to improve efficiency in [Noa09] is to remove public-key encryption and signatures. However, the improved method relies on the

well-known Diffie-Hellman key agreement which has not only a comparable complexity but also requires to stick to a particular cryptographic assumption. In contrast, original RAKE is more flexible in that it uses generic public-key primitives for signatures and encryption and can be implemented under different cryptographic assumptions.

Chapter 7

Deployment

SWISH tries to solve a very pragmatic problem. It is therefore very important for such a mechanism to be usable in actual networks. SWISH has been implemented and deployed in working environments in order to demonstrate that it could be used at a large scale.

In this chapter, we first introduce our prototype implementation and discuss its deployment in corporate and home networks. Then we compare the performance of RAKE in terms of connection delay *w.r.t.* popular authentication methods and discuss potential scalability issues that may appear during the tunneling phase.

7.1 Implementation

The current SWISH implementation is composed of two blocks: the RAKE protocol and a tunneling mechanism. The former is used to authenticate each party and to derive keys that are used by the latter to establish authenticated — and optionally encrypted — tunnels.

We chose to implement RAKE as an EAP method and to include it inside the *hostap* [Mal] software. The tunneling phase uses IPsec to achieve its goals. This section explains these choices and how these parts are working together.

7.1.1 Overview of EAP

The *Extensible Authentication Protocol* (EAP) [ABV+04] is an authentication framework that aims at being used at low layers when IP is not required or has not been configured yet. It runs over data-link protocols such as Point-to-Point Protocol (PPP) or IEEE 802, e.g., Ethernet or WiFi. Over IEEE 802, EAP is encapsulated within EAPOL (designed for IEEE 802.1X [IEE04]) to be used over Ethernet or WiFi (IEEE 802.11 [IEE07]). In practice, the most important usage of EAP nowadays is for WiFi. The WPA- and WPA2-enterprise protocols proposed by the WiFi alliance are based on EAP.

EAP is an extensible protocol, which indicates that it only offers the framework but no authentication in itself. The authentication as well as the key exchange, used for encrypting the WiFi channel if any, are managed by special algorithms on top of EAP, called methods. Many EAP methods have been designed; some have been standardized at IETF [FBW08,SAH08, ABV+04,HS06,AH06,CWMSZ07], some others are vendor specific [TNC05, SMBS04].

RAKE has been designed to authenticate mobile users on WiFi networks. Since we had to rely anyway on the IEEE 802.11 layer, using the IEEE 802.1X and EAP framework quickly seemed to be the best solution to allow RAKE to be included in existing architectures.

The 802.1X/EAP protocol is depicted on Figure 7.1. This represents a common enterprise scenario where the access point cannot identify the users by itself but relies on the enterprise's authentication server to do so. In EAP, the mobile device is called supplicant or peer, the authentication server is called the authenticator or the EAP server. There is no home nor visited networks as we are not in a roaming environment yet. When the mobile device is associated with the access point at the 802.11 layer, the device sends an *EAPOL start* message. At ①, the port of the access point is blocked, which implies that the access point only accepts EAP messages from this mobile device. EAP works by requests — sent by the authenticator — followed by responses — sent by the supplicant. The first EAP request is always an *identity request*. This identity is used by the authenticator to decide which EAP method it will use with the connected device. At ②, the

access point does not know how to authenticate the user and so forwards the identity response to the authentication server inside a *RADIUS Access request* message. Upon reception of this message (③), the authentication server determines which EAP method to use with the mobile device and creates an EAP state-machine for this method and this user. Then several EAP requests and responses are exchanged between the server and the mobile device. The number of exchanges depends on the particular method. Once the authentication and key exchange succeed (④), the EAP state machine is flushed and an *EAP Success* message is sent. The RADIUS message also contains the negotiated key so that the access point is able to derive keys to encrypt the wireless channel. When the access point receives the *EAP Success*, it transmits it to the mobile device and sets its port in the *authorized* state. Finally, some EAPOL and 802.11 packets are exchanged to complete the association. Note that even if we only mention wireless in this section, this works nearly exactly in the same way for a wired connection using 802.1X.

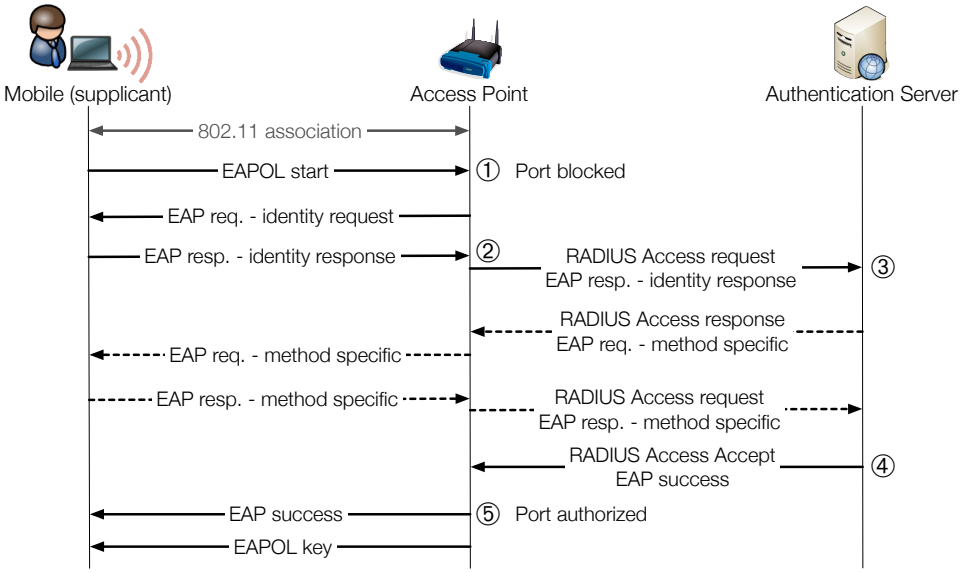


Figure 7.1: The EAP protocol with an authentication server

On the implementation side, the EAP standard defines an *EAP Switch Model* for interaction between EAP and its methods [VEPO05]. The EAP switches (one for the peer, one for the server) control the negotiation of EAP methods, and success or failure. In the access point, the switch also aims

to decide whether a EAP packet must be treated locally or encapsulated to a backend EAP server. Hooks between the EAP switch and any EAP method are clearly defined. For the peer, the EAP switch can request to initiate the state machine, to delete it, to check whether an incoming request is correct, to respond to one, and to retrieve the negotiated EAP key. For the authenticator, the EAP switch performs similar actions but asks for the next *EAP request* generation instead of the response.

7.1.2 Implementation of RAKE in hostap

Our new EAP method has been integrated in *hostap* [Mal] which provides WiFi drivers, an authentication daemon ($\mathcal{V}$) and a WPA supplicant ($\mathcal{M}$). We modified the latter two to support EAP-RAKE. The message exchange between $\mathcal{V}$ and $\mathcal{H}$ is not covered by *hostap* directly since EAP does not provide any way to include a third party in addition to a peer and an authenticator. We thus defined a *RAKE client* ($\mathcal{V}$) and a *RAKE server* ($\mathcal{H}$) that are able to communicate together over UDP.

The architecture of our implementation within *hostap* is shown on Figure 7.2. The *EAP peer switch*, *EAP server switch* and the *hostap library* were already implemented in *hostap*, the other tiles are RAKE specific. *EAP-RAKE peer* implements $\mathcal{M}$'s behavior, the *EAP-RAKE server* implements the visited network, and *RAKEd/rake* the home network. These three modules deal with the parsing and the creation of the RAKE messages according to the current state-machine status.

The layer containing EAP switches, *RAKE client* and *server* communicates with the upper layer via aforementioned hooks. *RAKEd* is the daemon that deal with the transport layer communication and initiates new sessions in $\mathcal{H}$.

We also defined a large library which provides the payload parsing, message assembly, configuration parsing, payload data computations, ... Each payload type has its own module and each data is treated as object. This design allows clear and modular source code. For the cryptographic function calls, we support the use of either the *hostap* internal implementations or OpenSSL.

Agile development methods have been followed for our implementation.

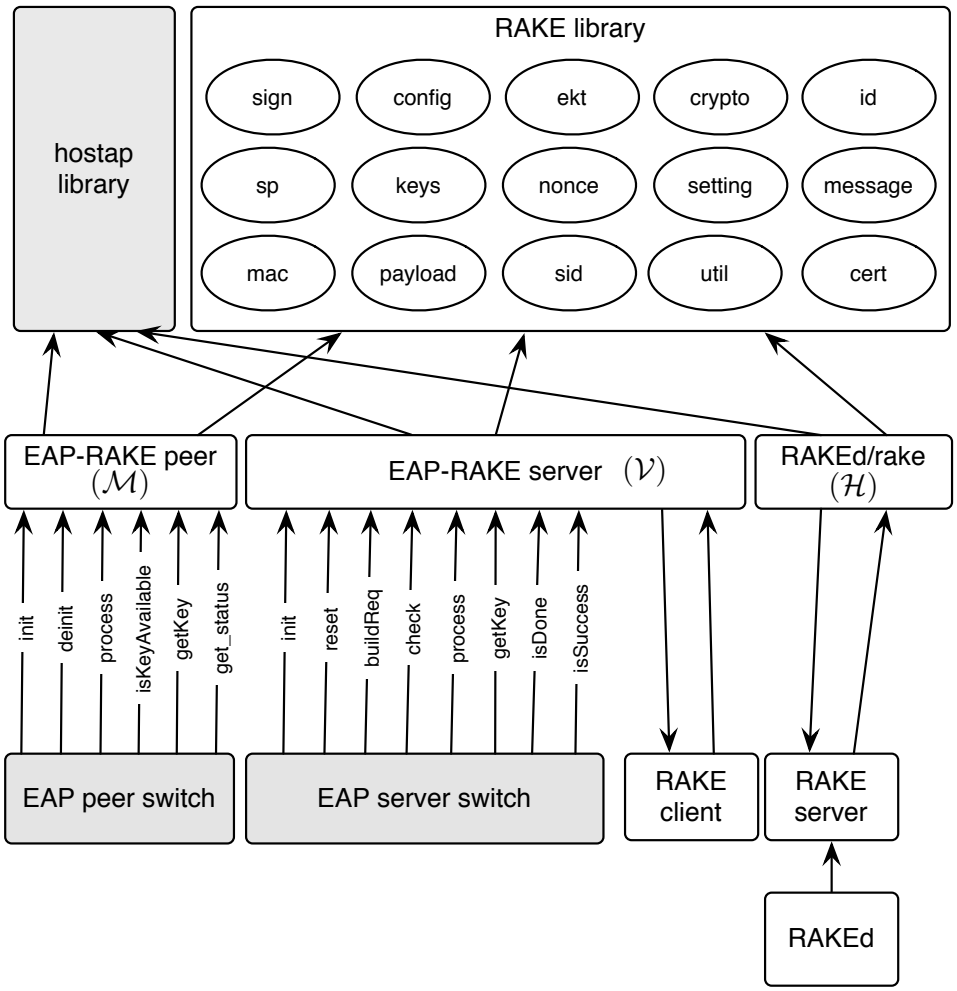


Figure 7.2: Design of the RAKE implementation in *hostap*

Unit and black-box tests have been written for most parts of the software. They are all re-run each time we develop a small new iteration. We have currently implemented a total of 60 files for about 12,500 lines of C, library and test cases included.

7.1.3 Tunneling

The tunneling phase of SWISH requires:

- The tunneling of all data sent by $\mathcal{M}$, from the first IP hop in the visited network, to $\mathcal{H}$.

- Authentication between $\mathcal{V}$ and $\mathcal{H}$ to avoid an attacker that would inject traffic in data flow.
- Optional encryption between $\mathcal{M}$ and $\mathcal{H}$ if they do not trust $\mathcal{V}$.

These requirements can be fulfilled by creating IPsec tunnels. IPsec is a security architecture that have been designed for communications relying on the IP (both IPv4 and IPv6) layer [KS05]. This architecture defines different *security protocols*, *security associations*, the *key management* and the *cryptographic algorithms* that can be used.

Two *security protocols* are defined, AH [Ken05a] and ESP [Ken05b]. Both operate between the network (IP) and the transport (TCP, UDP, ...) layers[1] and secure all upper layers. The Authentication Header (AH) protocol can only be used for authentication. The packet payload as well as the constant IP header fields are included in the computation. The Encapsulating Security Payload (ESP) protocol can perform authentication, encryption or both. AH and ESP can be used in transport or tunnel mode. Actually, the tunnel mode corresponds to a protected IP-over-IP tunnel and should be used when one of both ends does not correspond to the ends of the upper-layer connections.

For SWISH, two different channel have to be secured, the $\mathcal{V}$-$\mathcal{H}$ and the $\mathcal{M}$-$\mathcal{H}$ ones. Since neither of them ends at both sides at an end-host, they must be operated in tunnel mode. The first one only needs AH. The second one uses ESP for both encryption and authentication.

In each IPsec host, a *Security Association* (SA) database defines which IPsec policy to apply for which packets. A SA defines the protocol (AH or ESP), its operation mode, the *Security Parameter Index* (SPI), the cryptographic algorithm and the keys. The SPI is an index exchanged in clear in ESP and AH that can be used to identify the flows. For one tunnel, there are two SA, one for each direction. In SWISH, the SA are exchanged by using the RAKE protocols, in S^{Tun} and S^{TEnc} payloads.

In the IPsec architecture, the *key management* can be done either via the Internet Key Exchange (IKE) protocol or manually. IKE [KHNE10] is also used to negotiate the *cryptographic algorithms* supported by each party. In SWISH, the RAKE protocol includes key exchange. Additionally,

[1] Actually AH and ESP can also operate as an IPv6 Extension.

we can rely on RAKE to negotiate the cryptographic algorithms that will be used during the tunneling phase. EAP-RAKE totally replaces IKE in this context: tunnel type, algorithm and keys are configured by the SWISH process directly as if they were defined manually.

In practice, we use the IPsec implementation from the *Openswan* project [Xel]. Addition and removal from the SA database are performed by interacting directly with the API of the kernel module.

7.1.4 The SWISH architecture

Until now, only independent tiles of SWISH have been explained but not how they communicate together. This architecture is depicted on Figure 7.3.

First, we must note that the entity denoted $\mathcal{V}$ until now may actually be physically composed of three distinct entities. First, the access point is the device to which the mobile user connects. As already mentioned in Section 7.1.1, the access point, in enterprise configurations, may not be able to perform authentication by itself. In this case, it only relays the EAP packets to an authentication server. Once 802.1X has completed the authentication, the access point keeps a state until the mobile device disconnects. In this case, the *authentication server* is the authenticator that implements the EAP-RAKE protocol. Once the authentication has been completed, this server flushes all data related to this authentication. Finally, the tunnel start-point may be a different host. The latter should be on the same LAN as $\mathcal{M}$ and should be its default gateway to the Internet. The motivations of deploying SWISH on several servers could be to centralize data (e.g., the authentication), for scalability or for performance reasons. In the home network, the authenticator may also be different from the tunnel end-point.

As seen in Figure 7.3, the RAKE protocol runs between the mobile device and the authenticators of $\mathcal{V}$ and $\mathcal{H}$. Once completed, the EAP framework in $\mathcal{M}$ and $\mathcal{V}$ informs the lower layers of the negotiated key. At the same time, the RAKE modules pass the SA related to the client to the SWISH monitors, managing the tunnels. Once done, the RAKE process can terminate. Three connections must be kept alive: the 802.1X/WiFi connection between the mobile and the access point, the authenticated tunnel between $\mathcal{V}$ and $\mathcal{H}$, the encrypted tunnel between $\mathcal{M}$ and $\mathcal{H}$. The first one is managed by the

802.11 architecture. The latter two are managed by *SWISH monitors* which are responsible for the SA database and send keep-alive between entities to ensure that they are still reachable.

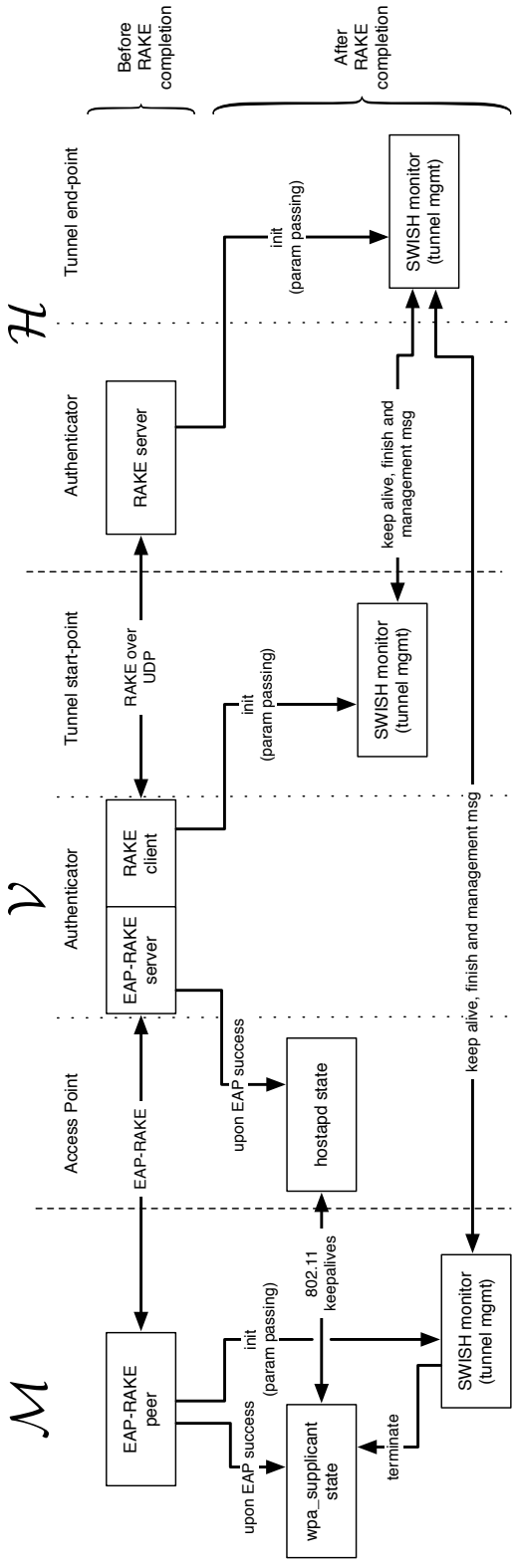


Figure 7.3: SWISH Global Architecture

7.2 Deployment Scenarios

A key issue for any WiFi sharing solution is that it should be possible to deploy it in existing networks as an additional service. In this section, we discuss the deployment of our prototype in corporate and home networks.

7.2.1 Deploying SWISH in Enterprise Networks

A first deployment scenario consists in using SWISH between a set of companies or institutions that want to allow their users to roam securely, and to offer an Internet access to their visitors. This deployment has similar incentives as Eduroam between educational institutions [Edu].

Today's enterprise networks typically use simple WiFi access points managed by a central controller [GJ09]. These simple access points can advertise multiple SSIDs, but all the authentication is performed by the controller. To evaluate SWISH in a real network, we deployed it in two campus networks (Figure 7.4). The first network, located at UCLouvain, acts as the visited network. It is composed of *Cisco Aironet* WiFi access points managed by a *Cisco WiSM* controller. The WiFi access points advertise a specific RAKE SSID and the controller forwards all authentication requests received on this SSID to our authentication server over RADIUS. The WiFi controller and the access points were not modified to support RAKE thanks to the EAP framework and RADIUS. The only required changes were a few lines of configuration on the WiFi controller.

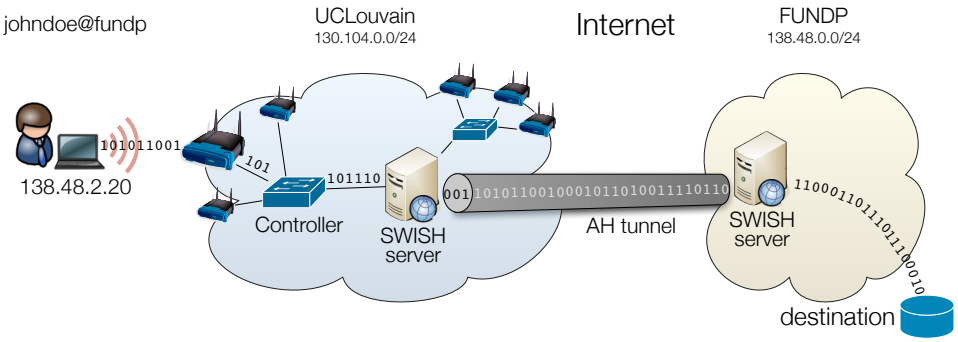


Figure 7.4: SWISH deployment architecture between the UCLouvain and FUNDP networks

To be able to create the required tunnels, the SWISH server must act

as $\mathcal{M}$'s gateway to the Internet. For this, we configured the controller to perform layer-2 bridging between the SWISH-enabled SSID and the SWISH server. In a large network, it would be possible to spread the load over multiple SWISH servers.

The second campus network, located on the premises of FUNDP, serves as a home network. The authentication and the tunnel end-point are deployed on a single SWISH server running Linux and our software.

Within this infrastructure (Figure 7.4), we used a Linux computer running EAP-RAKE to connect to the RAKE-enabled SSID in UCLouvain and obtained access to the Internet through the FUNDP network. As expected, since FUNDP and UCLouvain are well connected through the BELNET network, no latency was observable.

7.2.2 Deploying SWISH in User-Provided Networks

As indicated in Section 5.2, WiFi sharing is also popular among home users. SWISH can also be deployed in these environments. Our implementation runs on the OpenWRT Linux distribution that is used on several types of Linux-based DSL routers. In this case, the modified router can serve as both $\mathcal{V}$ — as an access point, authenticator, and tunnel start-point — for visitors, and $\mathcal{H}$ — as an authenticator and tunnel end-point — for roaming householders.

However, letting each user deploy SWISH on his own is not the best deployment strategy. Using a DSL router as $\mathcal{H}$ would provide limited performance given its low bandwidth and relatively high delays of the access links. A better strategy would be for DSL or CATV ISPs to install SWISH on their set-top boxes and to maintain a set of SWISH servers in their backbone that can be used by their users. Users would benefit from high performance while the ISP would have a large WiFi coverage thanks to its users. Such a service would probably be very interesting for network operators that do not already offer wireless services.

7.3 Performance

In this section, we first compare the performance of RAKE in terms of connection delay *w.r.t.* popular authentication methods. We then discuss potential scalability issues that may appear during the tunneling phase.

Our testbed was composed of three devices that play the roles of $\mathcal{M}$, $\mathcal{H}$ and $\mathcal{V}$. $\mathcal{M}$ was directly connected to $\mathcal{H}$ which was directly connected to $\mathcal{V}$ via Gigabit Ethernet. For regular measurements, the three devices were 2.6 Ghz Pentium 4 and 1 GBytes RAM computers running Linux and equipped with Gigabit-Ethernet cards. These were able to forward 512-byte packets at a rate of up to 237 Mbps. For the low-end access point measurement, we used as $\mathcal{H}$ an Asus WL-500gp access-point (Broadcom BCM47XX 266 Mhz CPU, 32 MBytes of RAM) running OpenWRT 8.09. It was able to forward 512-byte packets up to 33 Mbps. Finally, for the smartphone measurements, we used as $\mathcal{M}$ a Nokia N900 (ARM Cortex-A8 600 MHz CPU, 256 MBytes of RAM) running Maemo 5, a Linux-based OS.

7.3.1 RAKE Authentication Delay

We first evaluate the authentication delay of RAKE and compare it with two popular EAP-based protocols: EAP-TTLS [FBW08] and EAP-PEAP [PSS+04]. This is an optimistic environment from an authentication delay viewpoint as the delay between the different hosts is lower than a real deployment. We measured the authentication delay as the delay between the end of the EAP discovery phase that is common in the three schemes and the EAP-Success message that authorizes $\mathcal{M}$. In our lab, an EAP-TTLS/MD5 authentication was performed in 12.9 milliseconds while an EAP-PEAP/MSCHAPV2 authentication required 15.5 milliseconds. Figure 7.5 provides additional details by showing the execution time used by $\mathcal{V}$ and $\mathcal{M}$. This execution time was computed by analyzing a packet trace collected on all the participating machines. The difference between the authentication and the execution time is the network delay between the hosts.

Despite the fact that RAKE involves three parties, the authentication delay for $\mathcal{M}$ in our lab was only 17.4 milliseconds [2]. Figure 7.5 shows that

[2] It should be noted that the actual delay before success for $\mathcal{H}$ is slightly higher. $\mathcal{H}$ has indeed to process the *MA2* message once $\mathcal{M}$ and $\mathcal{V}$ have already completed.

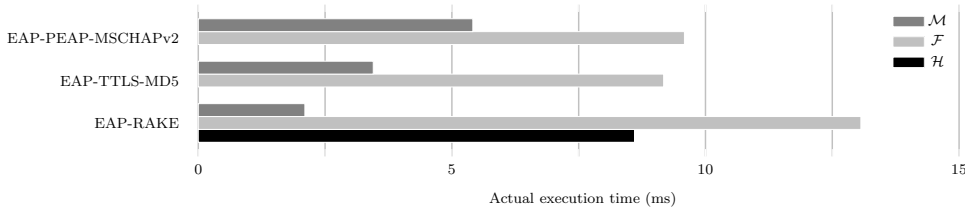


Figure 7.5: In comparison with other popular EAP authentication methods, our implementation of EAP-RAKE performs well. It is lighter for the mobile device but heavier for the visited network. The cost for the home network that has to authenticate both $\mathcal{M}$ and $\mathcal{V}$ is similar than EAP server authentication in the popular methods.

$\mathcal{M}$ spends only slightly more than 2 milliseconds to process the RAKE messages. This is lower than the processing time required by the two other authentication methods. We fairly mention that the actual authentication duration of RAKE may be higher than the other local authentication methods as RAKE needs to send messages over the Internet. Still, methods such as Eduroam [Edu] using EAP-TTLS [FBW08] for roaming also require to authenticate the mobile user in his home network, which requires six Internet round-trip times in our campus network. In RAKE, most of the execution time is spent on $\mathcal{V}$. A closer look at the measurements reveals that the longest computation time are the processing of the *HA1* message and the generation of the *MA2* message, which involve a decryption and a signature verification. Likewise, for $\mathcal{H}$ the longest execution time is spent generating the *HA1* message. Nevertheless the total execution time remains low. It should be noted that as of this writing, RAKE has not yet been optimized to reduce its computation time.

We also performed measurements with lower end devices. Figure 7.6 shows the execution time for $\mathcal{M}$ on a smartphone. First, we can see that, with 9.9 milliseconds of computation time, RAKE is much faster than other popular EAP methods on the client device. Second, unlike other evaluated EAP methods, the computation time on client side is fully independent from the key size as our protocol relieves the — presumably resource-constrained — mobile device from any costly asymmetric cryptography computation. This result shows clearly that our goal of having an authentication scheme that can be easily supported by mobile devices such as smart phones or notebooks is achieved. It also means that RAKE is appropriate for han-

dovers between different networks (both IP networks or between GSM/3G and WiFi).

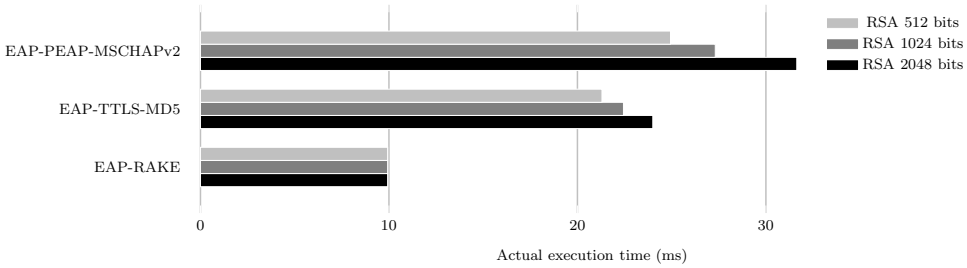


Figure 7.6: Actual computational time on $\mathcal{M}$ measured with a Nokia N900. The computation time on a light mobile device for EAP-RAKE remains low and is not increasing with the size of asymmetric keys as asymmetric encryption is not used in the mobile devices.

7.3.2 SWISH Tunneling Scalability

A campus network implementing SWISH could potentially need to support a large number of mobile devices at the same time, e.g., during a conference organized on the campus. Once the mobile devices have been authenticated, the main cost of SWISH are the IPsec tunnels used by $\mathcal{V}$.

In order to determine the tunneling forwarding limit, we performed measurements with several clients sending unidirectional constant UDP traffic to a tunnel start-point. The client was running D-ITG [BDP07] and generated a 1 Mbps stream composed of 512-byte packets. The client was connected via a Gigabit-Ethernet link to the tunnel-server that sent the IPsec packets over a second Gigabit-Ethernet interface. The server used for the measurements is far below current server's specifications. We also performed measurements by using a low-end standalone access point as tunnel start-point server.

Figure 7.7 provides, for the visited network, the bandwidth (or the number of 1 Mbps-clients) that can be supported by the Pentium server and the access-point without losing packets. The Pentium server is able to encapsulate IPsec packets at up to 161 Mbps using AH with HMAC-SHA1 and at up to 149 Mbps using AH with HMAC-SHA256. In contrast, the impact of AH on the access point is more significant since it already starts to drop packets above 9 Mbps using AH with HMAC-SHA1. The latter bandwidth is more than needed for visitors as 8.2 Mbps is observed as the mean residen-

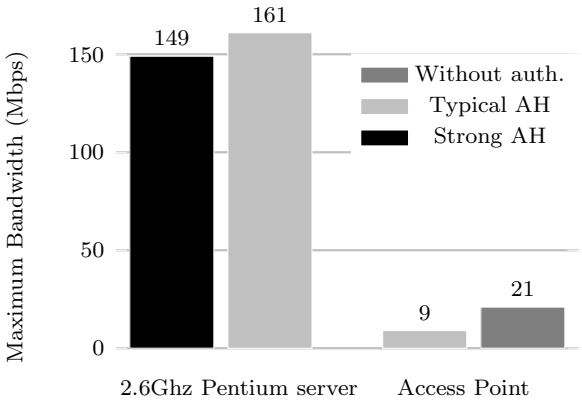


Figure 7.7: The cost of tunneling for $\mathcal{V}$ is rather low. On an outdated Pentium server, tunneling is performed at more than 150 Mbps. On a low-end access point, a bandwidth of 9 Mbps is attained with AH, which is above the typical residential bandwidth. Offloading of the AH authentication on a dedicated server is a way to allow much more visitors to use a single access point

tial download speed around the globe[3]. However, if better performances are required, the access points could also forward data to dedicated tunneling servers in their own ISP network using an unprotected tunnel (as explained in Section 7.2.2). Figure 7.7 shows that it is possible to forward traffic at 21 Mbps through such an unprotected GRE tunnel using the same access point.

To evaluate the scalability for $\mathcal{H}$, we used three scenarios: only using an authenticated tunnel (HMAC-SHA1); using both an authenticated tunnel (HMAC-SHA1) and typical ESP encryption with the mobile (HMAC-SHA1+AES128-CBC); using both an authenticated tunnel (HMAC-SHA256) and strong ESP encryption with the mobile (HMAC-SHA256+AES256-CBC). Figure 7.8 shows the maximum throughput obtained on $\mathcal{H}$. It shows that we can reach 161 Mbps with only an AH tunnel. If encryption is enabled, when using typical AH+ESP (resp. strong AH+ESP security parameters), the server's limits are reached when the total customers' bandwidth reaches 79 Mbps (resp. 69 Mbps). This implies that networks that have to handle a large number of customers should dimension their servers accord-

[3]Source: http://www.netindex.com/. 8.22 Mbps for download and 2.31 Mbps for upload, on Nov 10, 2010. The value is the rolling mean throughput in Mbps over the past 30 days where the mean distance between the client and the server is less than 300 miles.

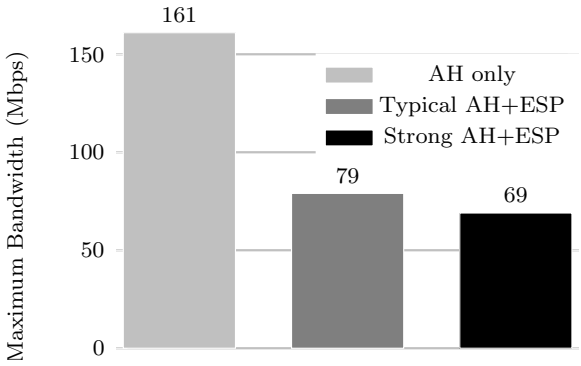


Figure 7.8: The cost of using an encrypted tunnel is more important for $\mathcal{H}$. On the same server as for $\mathcal{V}$, the maximum bandwidth is 161 Mbps for AH only and 79 Mbps if both AH and ESP are used with typical key sizes.

ingly. However, we should expect that server-class computers using IPsec hardware acceleration would handle a much higher bandwidth. Moreover, we remind that the strong security settings are far beyond current security recommendations for such tunneling [Ecr10].

Chapter 8

Conclusion

The ubiquity of WiFi mobile devices encourages users to share their WiFi Internet access. Popular WiFi sharing solutions often address some of the security issues for the mobile user but often overlook the security of the shared network. It is now easy to find — in the press — examples of abuse made on unprotected WiFi networks [Shi03, Aco07, Dic09, Par07]. Additionally, more and more countries are currently fighting against Internet sharing which cannot guarantee user identification, for the sake of anti-terrorism or copyright infringement [LDN10, BBC10, Ski10, Mey10].

SWISH simultaneously meets the concerns of those who want to provide or use shared WiFi Internet access. It provides the security guarantees required by network owners while respecting both the security and the privacy of mobile users. This is achieved by forwarding the packets of the mobile user through a tunnel to his home network using a provably secure cryptographic protocol. In addition to the core mechanism, the SWISH adaptive accounting protocol allows the visited network to safely charge for the utilization of the shared Internet access. The privacy extension prevents the visited network from tracing visitors while allowing the visitors' home networks to identify its own users.

We showed in the previous chapter how SWISH can easily be deployed in both enterprise and home WiFi networks. The prototype we have developed is working in these two environments. Our measurements with a non-optimized prototype indicate that the performance of SWISH is comparable to the performance of deployed authentication schemes that do not

provide the same security features as SWISH.

WiFi sharing will probably play soon an important role for mobile operators as a complement to other — more expensive — mobile data solutions such as 3G or LTE. However, we think that offloading can only be done on WiFi if we can rely on a good coverage and if security is ensured. Solutions such as SWISH, if deployed by large operators or user communities, fulfill these goals and may help the convergence of all mobile technologies. Some additional features could still be investigated to improve our proposal, for instance, fast roaming (e.g., from a WiFi to WiFi or from 3G to WiFi in a city), optimized performance, load balancing between tunneling servers, ...

General Conclusion

This thesis was aimed at making renumbering and sharing of IP networks safer and easier. It was structured in two parts, the first one focused on IPv6 site renumbering, and the second one on WiFi sharing.

In the first part of the thesis, we explained that being able to renumber an IPv6 network is a key operation for the scalability of the future Internet. Then, we showed that that renumbering, even if sometimes complex, can be partially automated. Firstly, we described a new mechanism for address allocation and propagation. This mechanism automates a part of the process while ensuring strong security. Secondly, we showed that it is possible to ease the update of configuration files during a renumbering by using macros and some specific rules for transition.

In the second part of the thesis, a Secure WIfi SHaring solution, SWISH. Its design and implementation are key contributions of this thesis. SWISH is compatible with existing standards and allows a network to share its Internet connection to visitors without any risk from misbehaviors. The visitors are also fully protected against a malicious network which would like to sniff or alter the content they sent. SWISH is composed of an authentication and key exchange protocol, called RAKE which is implemented above the EAP protocol, and a tunneling phase implemented by using IPsec tunnels.

The growing number of mobile devices connected to the Internet slowly transforms the existing networking models and makes new needs appear. This thesis addressed two important issues for the current and the future Internet. The approach that was chosen is definitely practical and several tools have been developed to demonstrate the applicability of our solutions. We think that these solutions could contribute to the upcoming evolutions of network management and security.

Bibliography

[3GP05] 3GPP Technical Specification. Telecommunication management; Charging management; Charging architecture and principles. Technical Report TS 32.200, v.5.9.0, 3rd Generation Partnership Project, October 2005.

[3GP06] 3GPP Technical Specification. Telecommunication management; Charging management; Charging architecture and principles. Technical Report TS 32.225, v.5.11.0, 3rd Generation Partnership Project, March 2006.

[3GP07] 3GPP Technical Specification. Tech. Spec. Group Core Network and Terminals; Handover Procedures (Release 7). Technical Report TS 33.102, v.7.0.0, 3rd Generation Partnership Project, March 2007.

[3GP08] 3GPP Technical Specification. Tech. Spec. Group Services and System Aspects; 3G Security; Security Architecture. Technical Report TS 33.102, v.8.0.0, 3rd Generation Partnership Project, June 2008.

[3GP10] 3GPP Technical Specification. Telecommunication management; Charging management; Charging architecture and principles. Technical Report TS 32.240, v.10.0.0, 3rd Generation Partnership Project, December 2010.

[AAL+05] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. DNS Security Introduction and Requirements. RFC 4033, Internet Engineering Task Force, March 2005.

[ABS03] B. Anton, B. Bullock, , and J. Short. Best Current Practices for Wireless Internet Service Provider (WISP) Roaming. Wi-Fi Alliance - Wireless ISP Roaming (WISPr), February 2003.

[ABV+04] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz. Extensible Authentication Protocol (EAP). RFC 3748, Internet Engineering Task Force, June 2004.

[Aco07] B. Acohido. Public Wi-Fi use raises hacking risk. http://www.usatoday.com/tech/wireless/2007-08-06-wifi-hot-spots_N.htm [Retrieved 2010-10-25], August 2007.

[AH06] J. Arkko and H. Haverinen. Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA). RFC 4187, Internet Engineering Task Force, January 2006.

[AKZN05] J. Arkko, J. Kempf, B. Zill, and P. Nikander. SEcure Neighbor Discovery (SEND). RFC 3971, Internet Engineering Task Force, March 2005.

[AMT07] L. Andersson, I. Minei, and B. Thomas. LDP Specification. RFC 5036, Internet Engineering Task Force, October 2007.

[APN10] APNIC. IPv6 Address Allocation and Assignment Policy. "apnic-089", July 2010.

[Atk10] R. Atkinson. ILNP Concept of Operations. Internet Draft "draft-rja-ilnp-intro-04", Internet Engineering Task Force, June 2010.

[Aur05] T. Aura. Cryptographically Generated Addresses (CGA). RFC 3972, Internet Engineering Task Force, March 2005.

[Aus02] R. Austein. Tradeoffs in domain name system (DNS) support for internet protocol version 6 (IPv6). RFC 3364, Internet Engineering Task Force, August 2002.

[Bac08] E. Baccelli. Address Autoconfiguration for MANET: Terminology and Problem Statement. Internet Draft "draft-ietf-autoconf-statement-04", Internet Engineering Task Force, February 2008.

[BBC10] Big Irish crackdown on net piracy. http://www.bbc.co.uk/news/10152623 [Retrieved 2010-10-25], May 2010.

[BCF06] F. Beck, I. Chrisment, and O. Festor. A Monitoring Approach for Safe IPv6 Renumbering. In *Proc. Int. Multi-Conference on Computing in the Global Information Technology (ICCGI)*, August 2006.

[BCK96] M. Bellare, R. Canetti, and H. Krawczyk. Keying Hash Functions for Message Authentication. In *CRYPTO '96*, volume 1109 of *LNCS*, pages 1–15. Springer, 1996.

[BCM10] C. Bernardos, M. Calderon, and H. Moustafa. Survey of IP address autoconfiguration mechanisms for MANETs. Internet Draft "draft-bernardos-manet-autoconf-survey-05", Internet Engineering Task Force, June 2010.

[BDP07] A. Botta, A. Dainotti, and A. Pescapè. Multi-Protocol and Multi-Platform Traffic Generation and Measurement. In *IEEE INFOCOM 2007 Demo Session*, volume 45, pages 526–532, May 2007.

[BDS+03] D. Balfanz, G. Durfee, N. Shankar, D. K. Smetters, J. Staddon, and H.-C. Wong. Secret Handshakes from Pairing-Based Key Agreements. In *Proc. IEEE Symp. on Security and Privacy*, pages 180–196, 2003.

[BEG+] K. Bayarou, M. Enzmann, E. Giessler, M. Haisch, B. Hunter, M. Ilyas, S. Rohr, and M. Schneider. Towards Certificate-Based Authentication for Future Mobile Communications. *Wireless Personal Communications*, 29(3).

[Bel06] M. Bellare. New Proofs for NMAC and HMAC: Security without Collision-Resistance. In *CRYPTO'06*, volume 4117 of *LNCS*, pages 602–619. Springer, 2006.

[Bel10] M. Belshe. A Client-Side Argument for Changing TCP Slow Start. http://www.chromium.org/spdy/An_Argument_For_Changing_TCP_Slow_Start.pdf [Retrieved 2010-10-25], January 2010.

[BFLN96] H. Berkowitz, P. Ferguson, W. Leland, and P. Nesser. Enterprise Renumbering: Experience and Information Solicitation. RFC 1916, Internet Engineering Task Force, February 1996.

[Bid05] H. Bidgoli. *Handbook of Information Security: Information Warfare, Social, Legal, and International Issues and Security Foundations*, volume 2, pages 83–93. John Wiley & Sons, 2005.

[BLD05] F. Baker, E. Lear, and R. Droms. Procedures for Renumbering an IPv6 Network without a Flag Day. RFC 4192, Internet Engineering Task Force, September 2005.

[BM95] S. Bradner and A. Mankin. The Recommendation for the IP Next Generation Protocol. RFC 1752, Internet Engineering Task Force, January 1995.

[BN00] M. Bellare and C. Namprempre. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In *ASIACRYPT'00*, volume 1976 of *LNCS*, pages 531–545. Springer, 2000.

[BN08] M. Bellare and C. Namprempre. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. *J. Cryptology*, 21(4):469–491, 2008.

[CAF10] B. Carpenter, R. Atkinson, and H. Flinck. Renumbering Still Needs Work. RFC 5887, Internet Engineering Task Force, May 2010.

[CBR03] W. R. Cheswick, S. M. Bellovin, and A. D. Rubin. *Firewalls and Internet Security: Repelling the Wily Hacker – 2nd Edition.* Addison-Wesley Professional, 2003.

[CFML08] R. Coltun, D. Ferguson, J. Moy, and A. Lindem. OSPF for IPv6. RFC 5340, Internet Engineering Task Force, July 2008.

[CFT05] G. Chelius, E. Fleury, and L. Toutain. No Administration Protocol (NAP) for IPv6 router auto-configuration. *Int. J. Internet Protocol Technology*, 1(2), 2005.

[CFV06] T. Chown, A. Ford, and S. Venaas. Things to think about when Renumbering an IPv6 network. Internet Draft "draft-chown-v6ops-renumber-thinkabout-05", Internet Engineering Task Force, September 2006.

[CGM04] G. Camarillo and M. A. Garcia-Martin. *The 3G IP Multimedia Subsystem (IMS).* John Wiley & Sons, 2004.

[CH00] M. Crawford and C. Huitema. DNS Extensions to Support IPv6 Address Aggregation and Renumbering. RFC 2874, Internet Engineering Task Force, July 2000.

[Chi] Chillispot - Open Source Wireless LAN Access Point Controller. http://www.chillispot.info/ [Retrieved 2010-10-25].

[Cis10] Cisco Wireless YouGov Survey Results. http://www.realwire.com/writeitfiles/Cisco%20Wireless%20Consumer.pdf [Retrieved 2010-10-25], 2010.

[CLG+03] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, and J. Arkko. Diameter Base Protocol. RFC 3588, Internet Engineering Task Force, September 2003.

[CR96] B. Carpenter and Y. Rekhter. Renumbering Needs Work. RFC 1900, Internet Engineering Task Force, February 1996.

[Cra00] M. Crawford. Router Renumbering for IPv6. RFC 2894, Internet Engineering Task Force, August 2000.

[CSM+09] B. Carr, O. Sury, J. Palet Martinez, A. Davidson, R. Evans, F. Yilmaz, and I. Wijte. IPv6 address allocation and assignment policy. "ripe-481", September 2009.

[CWMSZ07] N. Cam-Winget, D. McGrew, J. Salowey, and H. Zhou. The Flexible Authentication via Secure Tunneling Extensible Authentication Protocol Method (EAP-FAST). RFC 4851, Internet Engineering Task Force, May 2007.

[DBV+03] R. Droms, J. Bound, B. Volz, T. Lemon, C. Perkins, and M. Carney. Dynamic host configuration protocol for IPv6 (DHCPv6). RFC 3315, Internet Engineering Task Force, July 2003.

[Dev09] Devicescape Wi-Fi Report - Q2 2009. http://www.devicescape.com/assets/docs/DevicescapeQ22009Wi-FiReportFinal.pdf [Retrieved 2010-10-25], 2009.

[DH98] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460, Internet Engineering Task Force, December 1998.

[DH01] A. Durand and C. Huitema. The Host-Density Ratio for Address Assignment Efficiency: An update on the H ratio. RFC 3194, Internet Engineering Task Force, November 2001.

[Dic09] K. Dickerson. Illegal movie download forces shutdown of free Wi-Fi. http://www.coshoctontribune.com/article/20091109/UPDATES01/91109015 [Retrieved 2010-10-25], November 2009.

[Dra03] R. Draves. Default Address Selection for Internet Protocol version 6 (IPv6). RFC 3484, Internet Engineering Task Force, February 2003.

[Ecr10] Ecrypt Recommendations. http://www.keylength.com/en/3/ [Retrieved 2010-10-25], 2010.

[Edu] Eduroam. http://www.eduroam.org [Retrieved 2010-10-25].

[ETS08] ETSI Technical Specification. Digital Cellular Telecommunications System (Phase 2+); Security Related Network Function. Technical Report TS 100 929, v.8.6.0, ETSI, January 2008.

[FB97] P. Ferguson and H. Berkowitz. Network Renumbering Overview: Why would I want it and what is it anyway? RFC 2071, Internet Engineering Task Force, 1997.

[FBW08] P. Funk and S. Blake-Wilson. Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0). RFC 5281, Internet Engineering Task Force, August 2008.

[FFML10] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis. Locator/ID Separation Protocol (LISP). Internet Draft “draft-ietf-lisp-07”, Internet Engineering Task Force, April 2010.

[FHT10] S. Farrell, R. Housley, and S. Turner. An Internet Attribute Certificate Profile for Authorization. RFC 5755, Internet Engineering Task Force, January 2010.

[FLYV93] V. Fuller, T. Li, J. Yu, and K. Varadhan. Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy. RFC 1519, Internet Engineering Task Force, September 1993. Obsoleted by RFC 4632.

[FON] FON. http://www.fon.com [Retrieved 2010-10-25].

[FRH10] A. Ford, C. Raiciu, and M. Handley. TCP Extensions for Multipath Operation with Multiple Addresses. Internet Draft "draft-ietf-mptcp-multiaddressed-01", Internet Engineering Task Force, July 2010.

[GHM+05] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang. Refactoring Network Control and Management: A Case for the 4D Architecture. Technical Report CMU-CS-05-117, CMU CS, September 2005.

[GJ09] M. L. Gress and L. Johnson. *Deploying and Troubleshooting Cisco Wireless LAN Controllers*. Cisco Press, 2009.

[GLD+08] S. Gundavelli, K. Leung, V. Devarapalli, K. Chowdhury, and B. Patil. Proxy Mobile IPv6. RFC 5213, Internet Engineering Task Force, August 2008.

[GMHM03] M. Garcia-Martin, E. Henrikson, and D. Mills. Private Header (P-Header) Extensions to the Session Initiation Protocol (SIP) for the 3rd-Generation Partnership Project (3GPP). RFC 3455, Internet Engineering Task Force, January 2003.

[GPS+03] J. Gu, S. Park, O. Song, J. Lee, J. Nah, and S. V. Sohn. Mobile PKI: A PKI-Based Authentication Framework for the Next Generation Mobile Communications. In *ACISP'03*, volume 2727 of *LNCS*, pages 180–191. Springer, 2003.

[Gra07] P. Gralla. Don't fall victim to the 'Free Wi-Fi' scam. http://www.computerworld.com/s/article/9008399/Don_

t_fall_victim_to_the_Free_Wi_Fi_scam [Retrieved 2010-10-25], January 2007.

[GXT+08] S. Goldberg, D. Xiao, E. Tromer, B. Barak, and J. Rexford. Path-quality monitoring in the presence of adversaries. *SIGMETRICS Performance Evaluation Review*, 36(1), 2008.

[Hal05] R. V. Hale. WiFi Liability: Potential Legal Risks in Accessing and Operating Wireless Internet. *Santa Clara Computer and High Technology Law J.*, 21:543, 2005.

[HD06] R. Hinden and S. Deering. IP Version 6 Addressing Architecture. RFC 4291, Internet Engineering Task Force, February 2006.

[HGWW08] T. Heer, S. Götz, E. Weingärtner, and K. Wehrle. Secure Wi-Fi Sharing on Global Scales. In *Proc. of 15th International Conference on Telecommunication (ICT '08)*, July 2008.

[HH05] R. Hinden and B. Haberman. Unique Local IPv6 Unicast Addresses. RFC 4193, Internet Engineering Task Force, October 2005.

[Hin05] M. Hines. Worried about Wi-Fi security? http://news.cnet.com/Worried-about-Wi-Fi-security/2100-7347_3-5540969.html [Retrieved 2010-10-25], January 2005.

[HS05] H. Hasan and B. Stiller. Non-repudiation of Consumption of Mobile Internet Services with Privacy Support. In *Proc. IEEE Int. Conf. on Wireless & Mobile Computing, Networking & Communications (WiMob)*, 2005.

[HS06] H. Haverinen and J. Salowey. Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM). RFC 4186, Internet Engineering Task Force, January 2006.

[IEE97] IEEE. Guidelines for 64-bit Global Identifier (EUI-64) Registration Authority. Technical report, IEEE, March 1997.

[IEE04] *802.1x-2004 IEEE Standard for Local and metropolitan area networks – Port-Based Network Access Control*. IEEE, New York, NY, USA, December 2004. Revision of IEEE Std 802.1X-2001.

[IEE07] *IEEE Standard for Information Technology-Telecommunications and Information Exchange Between Systems-Local and Metropolitan Area Networks-Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. December 2007.

[II01] IAB and IESG. IAB/IESG recommendations on IPv6 address allocations to sites. RFC 3177, Internet Engineering Task Force, September 2001.

[Ile05] D. Ilett. 'Evil twin' could pose Wi-Fi threat. http://news.cnet.com/Evil-twin-could-pose-Wi-Fi-threat/2100-7349_3-5545355.html [Retrieved 2010-10-25], January 2005.

[Int10] Internet World Stats. Miniwatts Marketing Group. World Internet Users and Population Stats. http://www.internetworldstats.com/stats.htm [Retrieved 2010-10-25], September 2010.

[Jeo06] J. Jeong. IPv6 Host Configuration of DNS Server Information Approaches. RFC 4339, Internet Engineering Task Force, February 2006.

[JPBM10] J. Jeong, S. Park, L. Beloeil, and S. Madanapalli. IPv6 Router Advertisement Option for DNS Configuration. RFC 6106, Internet Engineering Task Force, November 2010.

[KBC+10] J. Korhonen, J. Bournelle, K. Chowdhury, A. Muhanna, and U. Meyer. Diameter Proxy Mobile IPv6: Mobile Access Gateway and Local Mobility Anchor Interaction with Diameter Server. RFC 5779, Internet Engineering Task Force, February 2010.

[KCH+03] S. Kim, H. Cho, H. Hahm, S. Lee, and M. S. Lee. Interoperability between UMTS and CDMA 2000 Networks. *IEEE Wireless Communications*, 10(1):22–28, 2003.

[Ken05a] S. Kent. IP Authentication Header. RFC 4302, Internet Engineering Task Force, December 2005.

[Ken05b] S. Kent. IP Encapsulating Security Payload (ESP). RFC 4303, Internet Engineering Task Force, December 2005.

[KHNE10] C. Kaufman, P. Hoffman, Y. Nir, and P. Eronen. Internet Key Exchange Protocol Version 2 (IKEv2). RFC 5996, Internet Engineering Task Force, September 2010.

[KMZ02] S. Kremer, O. Markowitch, and J. Zhou. An Intensive Survey of Fair Non-Repudiation Protocols. *Computer Communications*, 25:1606–1621, 2002.

[KPS03] C. Kaufman, R. Perlman, and B. Sommerfeld. DoS protection for UDP-based protocols. In *Proc. ACM conf. on Computer and communications security*, CCS '03, pages 2–7. ACM, 2003.

[KS05] S. Kent and K. Seo. Security Architecture for the Internet Protocol. RFC 4301, Internet Engineering Task Force, December 2005.

[Lam78] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, 1978.

[LDC+11] D. Leroy, G. Detal, J. Cathalo, M. Manulis, F. Koeune, and O. Bonaventure. SWISH: Secure WiFi Sharing. *Communications Networks - Special Issue "Network Convergence"*, 2011.

[LDN10] Fears for Wi-Fi access - Germans fine open network surfers. http://www.thelondondailynews.com/fears-wifi-access-germans-fine-open-network-surfers-p-4113.html [Retrieved 2010-10-25], May 2010.

[LKS04] C. Lynn, S. Kent, and K. Seo. X.509 Extensions for IP Addresses and AS Identifiers. RFC 3779, Internet Engineering Task Force, June 2004.

[LWI04] M. Long, C.-H. Wu, and J. D. Irwin. Localised Authentication for Inter-Network Roaming across Wireless LANs. *IEEE Proceedings Communications*, 151(5):496–500, 2004.

[Mal] J. Malinen. Host AP driver for Intersil Prism2/2.5/3, hostapd, and WPA Supplicant. http://hostap.epitest.fi/ [Retrieved 2010-10-25].

[MCW05] U. Meyer, J. Cordasco, and S. Wetzel. An Approach to Enhance Inter-Provider Roaming through Secret Sharing and its Application to WLANs. In *WMASH'05*, pages 1–13. ACM, 2005.

[Mes10] E. Messmer. How Wi-Fi attackers are poisoning Web browsers. http://www.networkworld.com/news/2010/020310-black-hat-wi-fi-attackers.html [Retrieved 2010-10-25], February 2010.

[Mey10] D. Meyer. Open Wi-Fi 'outlawed' by Digital Economy Bill. http://www.zdnet.co.uk/news/networking/2010/02/26/open-wi-fi-outlawed-by-digital-economy-bill-40057470/ [Retrieved 2010-10-25], February 2010.

[Mic09] Microsoft. How Internet Explorer uses the cache for DNS host entries. http://support.microsoft.com/default.aspx?scid=KB;en-us;263558 [Retrieved 2010-10-25], March 2009.

[MLK+08] M. Manulis, D. Leroy, F. Koeune, O. Bonaventure, and J.J. Quisquater. Authenticated Wireless Roaming via Tunnels: Making Mobile Guests Feel at Home. http://eprint.iacr.org/2008/382, 2008. The shorter version of this full paper appears at ASIACCS 2009. Copyright ACM. Last revised 16 Dec 2008.

[MMS+05] A. S. Merino, Y. Matsunaga, M. Shah, T. Suzuki, and R. H. Katz. Secure Authentication System for Public WLAN Roaming. *Mobile Networks and Applications*, 10(3):355–370, 2005.

[MN06] R. Moskowitz and P. Nikander. Host Identity Protocol (HIP) Architecture. RFC 4423, Internet Engineering Task Force, May 2006.

[MR10] R. K. Murugesan and S. Ramadass. A Hybrid Address Allocation Algorithm for IPv6. In *Proc. Int. Conf. on Network Security Applications (CNSA-2010)*, pages 509–517, 2010.

[MST94] R. Molva, D. Samfat, and G. Tsudik. Authentication of Mobile Users. *IEEE Network*, 8:26–34, 1994.

[MW04] U. Meyer and S. Wetzel. A Man-in-the-Middle Attack on UMTS. In *WiSe'04*, pages 90–97. ACM, October 2004.

[MWA02] R. Mahajan, D. Wetherall, and Tom Anderson. Understanding BGP misconfiguration. *SIGCOMM'02*, August 2002.

[MZF07] D. Meyer, L. Zhang, and K. Fall. Report from the IAB Workshop on Routing and Addressing. RFC 4984, Internet Engineering Task Force, September 2007.

[NB09] E. Nordmark and M. Bagnulo. Shim6: Level 3 Multihoming Shim Protocol for IPv6. RFC 5533, Internet Engineering Task Force, June 2009.

[NDK07] T. Narten, R. Draves, and S. Krishnan. Privacy Extensions for Stateless Address Autoconfiguration in IPv6. RFC 4941, Internet Engineering Task Force, September 2007.

[NHR10] T. Narten, G. Huston, and L. Roberts. IPv6 Address Assignment to End Sites. Internet Draft "draft-narten-ipv6-3177bis-48boundary-05", Internet Engineering Task Force, July 2010.

[NNSS07] T. Narten, E. Nordmark, W. Simpson, and H. Soliman. Neighbor discovery for IP Version 6 (IPv6). RFC 4861, Internet Engineering Task Force, September 2007.

[Noa09] A. Noack. Efficient Authenticated Wireless Roaming via Tunnels. In *Proc. QSHINE*, 2009.

[Nor06] North American Network Operators' Group. Nanog mailing list. http://www.merit.edu/mail.archives/nanog/ [Retrieved 2010-10-25], 2006.

[NRO10] NRO. Remaining IPv4 Address Space Drops Below 5%. http://www.nro.net/media/remaining-ipv4-address-below-5.html [Retrieved 2010-10-25], October 2010.

[NS78] R. M. Needham and M. D. Schroeder. Using Encryption for Authentication in Large Networks of Computers. *CACM*, 21(12):993–999, 1978.

[NS2] NS-2 - The Network Simulator. http://www.isi.edu/nsnam/ns/ [Retrieved 2010-10-25].

[Pal09] J. Palet. Provider Independent (PI) IPv6 Assignments for End User Organisations. Technical report, February 2009. RIPE Policy Proposal 2006-01, version 5.

[Par07] N. Paris. Two arrested over wifi theft. http://www.telegraph.co.uk/news/uknews/1548960/Two-arrested-over-wifi-theft.html [Retrieved 2010-10-25], April 2007.

[PAS+04] J. Pang, A. Akella, A. Shaikh, B. Krishnamurthy, and S. Seshan. On the Responsiveness of DNS-based Network Control. *IMC'04*, October 2004.

[Per02] C. Perkins. IP Mobility Support for IPv4 . RFC 3344, Internet Engineering Task Force, August 2002.

[Pos81] J. Postel. Internet Protocol. RFC 791, Internet Engineering Task Force, September 1981.

[PSS+04] A. Palekar, D. Simon, J. Salowey, H. Zhou, G. Zorn, and S. Josefsson. Protected EAP Protocol (PEAP) Version 2. Internet Draft "draft-josefsson-pppext-eap-tls-eap-10", Internet Engineering Task Force, October 2004.

[QH08] L. Quan and J. Heidemann. On the Characteristics and Reasons of Long-lived Internet Flows. In *Proc. of Internet Measurement Conference 2010 (IMC '10)*, November 2008.

[RK04] G. Rose and G. Koien. Access Security in CDMA2000, including a Comparison with UMTS Access Security. *IEEE Wireless Commmunications*, 11(1):19–25, August 2004.

[RL95] Y. Rekhter and T. Li. An Architecture for IPv6 Unicast Address Allocation. RFC 1887, Internet Engineering Task Force, December 1995.

[RS99] C. Rackoff and D. R. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In *CRYPTO'91*, volume 576 of *LNCS*, pages 433–444. Springer, 199.

[RSC+02] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261, Internet Engineering Task Force, June 2002.

[RSZ04] C. Ribeiro, F. Silva, and A. Zúquete. A Roaming Authentication Solution for Wifi using IPsec VPNs with Client Certificates. In *TERENA Networking Conf.*, 2004.

[RWRS00] C. Rigney, S. Willens, A. Rubens, and W. Simpson. Remote Authentication Dial In User Service (RADIUS). RFC 2865, Internet Engineering Task Force, June 2000.

[SAH08] D. Simon, B. Aboba, and R. Hurst. The EAP-TLS Authentication Protocol. RFC 5216, Internet Engineering Task Force, March 2008.

[SBG+03] L. Salgarelli, M. Buddhikot, J. Garay, S. Patel, and S. Miller. Efficient Authentication and Key Distribution in Wireless IP Networks. *IEEE Wireless Communications*, 10(6):52–61, 2003.

[SE01] P. Srisuresh and K. Egevang. Traditional IP Network Address Translator (Traditional NAT). RFC 3022, Internet Engineering Task Force, January 2001.

[Sha79] A. Shamir. How to Share a Secret. *Communications of the ACM*, 22(11):612–613, 1979.

[Sha10] J. Sharkey. At Hotels, Making Wi-Fi as Standard as a Bed. http://www.nytimes.com/2010/05/06/business/06CONNECT.html [Retrieved 2010-10-25], May 2010.

[Shi03] R. Shim. Wi-Fi arrest highlights security dangers. http://news.cnet.com/Wi-Fi-arrest-highlights-security-dangers/2100-1039_3-5112000.html [Retrieved 2010-10-25], November 2003.

[Ski10] C. Skinner. Police ask internet cafes to snoop on users. http://www.networkworld.com/news/2010/032610-police-ask-internet-cafes-to.html [Retrieved 2010-10-25], March 2010.

[SMBS04] S. Sundaralingham, D. Miller, A. Balinsky, and K. Sankar. *Cisco Wireless LAN Security*. Cisco Press, 2004.

[Sol04] H. Soliman. *Mobile IPv6: Mobility in a Wireless Internet*. Addison-Wesley Professional, 2004.

[SSC07] N. Sastry, K. Sollins, and J. Crowcroft. Architecting Citywide Ubiquitous Wi-Fi Access. In *Proc. HotNets-VI*, 2007.

[Ste07] R. Stewart. Stream Control Transmission Protocol. RFC 4960, Internet Engineering Task Force, September 2007.

[Swi06] K. Swiat. The Travelling Menace: Rogue Hotspots. *Computer Fraud & Security*, 2006(12):13–15, December 2006.

[SXT+07] R. Stewart, Q. Xie, M. Tuexen, S. Maruyama, and M. Kozuka. Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration. RFC 5061, Internet Engineering Task Force, September 2007.

[Sí09] D. M. Síthigh. Law in the Last Mile: Sharing Internet Access through WIFI. In *SCRIPTed 355*, volume 6, August 2009.

[TD03] O. Troan and R. Droms. IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6. RFC 3633, Internet Engineering Task Force, December 2003.

[TNC05] *TCG Trusted Network Connect – TNC IF-IMC*. TCG, May 2005. Specification Version 1.0 – Revision 3.

[TNJ07] S. Thomson, T. Narten, and T. Jinmei. IPv6 Stateless Address Autoconfiguration. RFC 4862, Internet Engineering Task Force, September 2007.

[UM10] J. Ubillos and Z. Ming. Name Based Sockets. Internet Draft "draft-ubillos-name-based-sockets-01", Internet Engineering Task Force, July 2010.

[Var08] M. Varsavsky. Why do some Foneros disconnect their Fonera WiFi router? http://english.martinvarsavsky.net/general/why-do-some-foneros-disconnect-their-fonera-wifi-router.html [Retrieved 2010-10-25], April 2008.

[VEPO05] J. Vollbrecht, P. Eronen, N. Petroni, and Y. Ohba. State Machines for Extensible Authentication Protocol (EAP) Peer and Authenticator. RFC 4137, Internet Engineering Task Force, August 2005.

[Vog09] C. Vogt. Six/One: A Solution for Routing and Addressing in IPv6. Internet Draft "draft-vogt-rrg-six-one-02", Internet Engineering Task Force, October 2009.

[VTRB97] P. Vixie, S. Thomson, Y. Rekhter, and J. Bound. Dynamic Updates in the Domain Name System (DNS UPDATE). RFC 2136, Internet Engineering Task Force, April 1997.

[Wex10] J. Wexler. Beware the rogue Wi-Fi access point in Windows 7. http://www.networkworld.com/newsletters/wireless/2010/022210wireless1.html [Retrieved 2010-10-25], February 2010.

[WG10] R. Wakikawa and S. Gundavelli. IPv4 Support for Proxy Mobile IPv6. RFC 5844, Internet Engineering Task Force, May 2010.

[WGP07] M. Wang, A. Goal, and B. Prabhakar. Tackling IPv6 Address Scalability from the Root. In *Proc. SIGCOMM 2007 Workshop "IPv6 and the Future of the Internet" (IPv6'07)*, August 2007.

[Xel] Xelerance Corporation. Openswan. http://www.openswan.org/ [Retrieved 2010-10-25].